mc
2
Die Mikrocomputer-Zeitschrift
7 DM · 60 öS · 7 sfr. · Februar 1987

Leserbefragung:
Atari 1040 ST zu gewinnen!

Systembus, EGA transparent:

PC-Technik

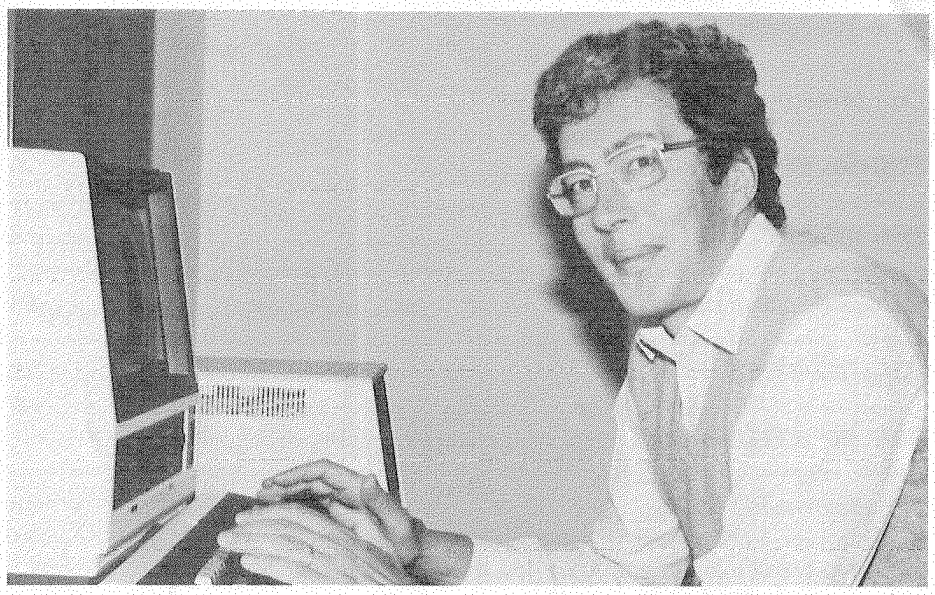
32 MByte zum Einstecken

1-MBit-EPROMs programmieren

Turbo-Pascal: DOS-Exec-Funktion Hercules-Grafik

Im Test: GfA-Basic für Atari ST Apple IIGS

Außerdem: CP/M-68 k, CP/M 3.0, mc-68-ECB, 68008-EMUF



Eine Million

Vor kurzem fand ein hübsches kleines Fest statt: Commodore gab sich die Ehre, den millionsten Verkauf des C-64 in der Bundesrepublik zu feiern. Eingeladen waren alle, die den Weg von Commodore in der Szene von den Anfängen her mitbeobachtet haben. Es gab allen Grund zu feiern, denn man muß sich nur vor Augen führen, daß Commodore hier in der Bundesrepublik das Zeitalter der Mikrocomputerei eröffnet hat. Der erste wirklich volksnahe Mikrocomputer, das war der PET 2001, der Personal Transactor, das Ding mit der Taschenrechner-Tastatur und dem Herzen eines Löwen, dem 6502. Irgendwie war dieser Mikrocomputer so typisch Commodore, wie es heute der C-64 ist und auch so erfolgreich. Ganze Scharen junger Enthusiasten haben auf diesem Rechner die Computerei erlernt, in allen Höhen und Tiefen. Und sie sahen ihren Ehrgeiz erst dann befriedigt, wenn sie bis in die letzten Geheimnisse des Betriebssystems eingestiegen waren, von Stackpointern und Variablenzeigern sprechen konnten, von Garbage-Routinen und Tastaturpuffern. mc hat damals mitgetan, denn mc hat schließlich die ROM-Listings veröffentlicht, auf daß jedermann durchblicke. Schöne Hackerzeiten waren das damals. Das fand seine Fortsetzung in Betrieben und Universitäten. Neben dem Apple II waren der PET und seine Nachfolger, die mit richtiger Tastatur ausgestatteten Commodore-30XX-Rechner, die bestdokumentierten Systeme, dank der Aufklärungsarbeit der Presse. Das lohnte sich für Commodore insofern, als diese Computer dann auch von Ingenieuren im Beruf in Betrieb genommen wurden, die auf dem PET aus Hobbygründen zu Hause eingestiegen waren. Natürlich spielte auch die besondere Eignung der Geräte für das Messen, Steuern und Regeln eine Rolle, sie hatten eine IEEE-Bus-Schnittstelle. Den Gedanken an einen Volkscomputer ließ Commodore nicht fallen. Es wurde der VC 20 geboren, das Gerät, das seine Softwarekonfiguration dynamisch nach Einstecken von Erweiterungen veränderte. Es war ein sehr preiswerter Computer mit gutem Markterfolg; aber im Zeichen der 64-KByte-RAM-Chips nicht auf Dauer up to date. Das änderte sich mit einem Schlag, als der C-64 geboren war, der die Vorteile der großen PET-Nachfolger mit den kleinen Spielgeeigneten so verband, daß wirklich ein Volkscomputer dabei herauskam. Gern hat mc viele Programme und Artikel dazu veröffentlicht. Denn wie bei jedem echten Commodore gibt es neben dem Handbuch noch viel zu erklären und zu berichten. Der C-64 war und ist so erfolgreich, daß ein japanischer Großangriff auf unsere Szene erfolglos blieb: Die MSX-Computer haben hierzulande nie so recht reüssiert, weil jedermann schon einen Commodore besaß, der genügend Farbfeuerwerk auf dem Bildschirm veranstalten konnte. Wir bei mc werden den C-64 nicht aus dem Auge verlieren, beglückwünschen Commodore zum Erfolg. Gespannt sind wir allerdings auch darauf, ob der Amiga nach einer Preissenkung, die eine Preishalbierung bedeutete, jetzt ebenso gut ins Geschäft kommt wie der C-64, der bald schon eine Preisdrittelung hinter sich hat.

Ihr

Ulrich Rohde

mc-inhalt

mc-kolumne

Eine Million 3

mc-briefe 6

mc-info 8

Spruch des Monats 38

mc-bücher 22

mc-grundlagen

Der gläserne PC-Bus 30
Bevor mit der Entwicklung spezieller Erweiterungskarten für den PC begonnen werden kann, muß man den Bus im Detail kennen

mc-soft

EGA: Print-Screen-Funktion 33
Ein paar Zeilen Assemblercode sorgen für verzerrungsarmen Ausdruck des Bildschirmspeichers

Telex mit PCs 34
Über 40 MS-DOS-Computermodelle dürfen jetzt als Telex-Endgerät eingesetzt werden

Vier Bildschirmseiten 35
Erweitern Sie die Anzeigekapazität des Bildschirms um das Vierfache

Startadresse von Apple-Binärdateien 38
DOS-Patch für Neugierige

Disk-Editor für CP/M-68k 40
Professionelles C-Programm für den mc-68000- und andere Computer zum Manipulieren von Diskettendateien

Programmieren in Assembler 44
Ein Software-Werkzeug für den Profi

Turbo-Pascal bedient Hercules-Karte 46
Wir stellen zahlreiche Prozeduren zum einfachen Einsatz einer Hercules-Karte mit einem PC vor

Taschenrechner mit DEBUG eingeben 49
Das Programm aus mc 12/86 für alle diejenigen, die nicht mit einem Assembler arbeiten

DOS-EXEC-Funktion von Turbo-Pascal aus 50
Von Turbo-Pascal aus werden beliebige andere DOS-Programme gestartet

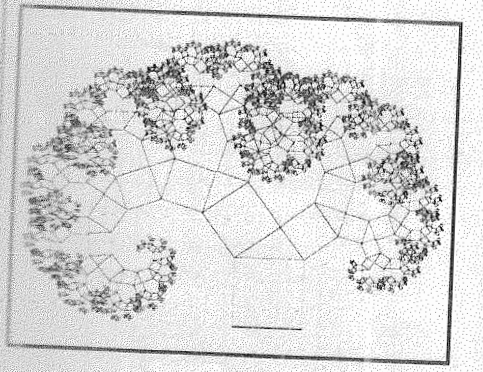
Grafik mit EGA 54
Im zweiten Teil unserer EGA-Serie beschreiben wir die EGA-Register – die Pforte zur PC-Grafikwelt

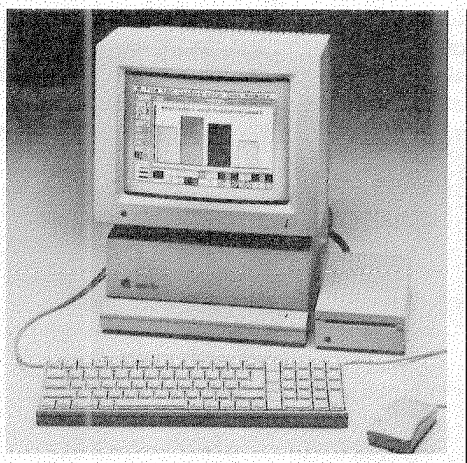
PC-Steckkarten

Ohne eine Erweiterungskarte kommt heute kaum ein PC aus. Neben Tests und Funktionsbeschreibungen von Steckkarten finden Sie einen Beitrag über den PC-Bus, damit selbstentwickelte Schaltungen auch in Ihrem PC funktionieren. Geballte Informationen ab **Seite 30**

PC-Grafik

So schön PC-Grafiken auch sind – ohne detailliertes Hintergrundwissen kann man sie nicht optimal einsetzen. Wir zeigen, wie EGA-Grafiken unverzerrt ausgedruckt und die EGA-Register richtig programmiert werden. Auch für Anwender der Herculeskarte haben wir einen Beitrag. Artikel über PC-Grafik ab **Seite 33**



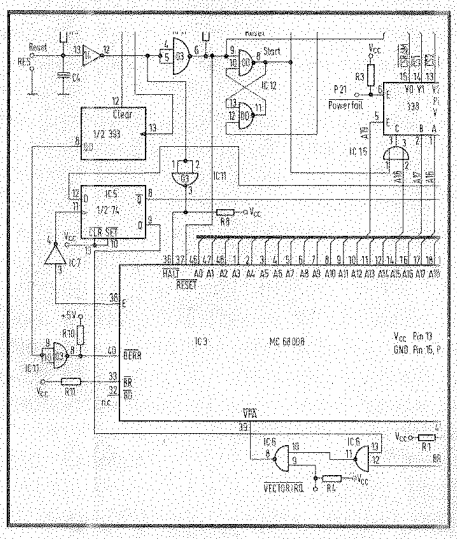


Apple-IIGS

Würde man das Heer der Apple-II-Anwender befragen, welche Fähigkeiten ein neues Modell haben sollte, würde wahrscheinlich ein anderer Computer dabei herauskommen als der jetzt erhältliche Apple-IIGS. Lesen Sie, was Apple's Neuer außer Kompatibilität zum Apple-II zu bieten hat. **Seite 92**

EMUF-08

Mit dem EMUF-08 stellen wir einen leistungsfähigen Einplatinencomputer vor, der besonders Anwender eines 68000er-Computers interessieren wird. Mit einem Arbeitsspeicher von 1 MByte beherrscht der EMUF-08 auch die komplizierteste Aufgabe schnell und zuverlässig. **Seite 101**



Sprites für den Apple-II 59
Mit Blockshapes lassen sich schnelle Grafiken einfacher realisieren

Turbo-Taktumschaltung 61
Der Wechsel der beiden Gangarten, die viele PCs bieten, funktioniert nach einer kleinen Änderung des Tastaturtreibers ohne Probleme

Eingabe unter Turbo-Pascal 62
Eine nützliche Prozedur vereinfacht die Texteingabe

Hex-Monitor für den mc-68-ECB 64
Ein komfortabler Maschinensprache-Monitor für den mc-68-ECB, der den Einsatz des 68000 im CP/M-System erlaubt

MS-DOS-Format unter CP/M 3.0 74
MS-DOS-Disketten können auch von CP/M-Systemen gelesen und beschrieben werden

Submit unter OS-9/68k 80
Auf Altbewährtes braucht man auch unter OS-9 nicht zu verzichten

mc-test

1-MBit-EPROMs programmieren 82
Ein interessantes Programmiergerät mit menügesteuerter Software für PCs

GfA-Basic 84
Mit diesem Interpreter für den Atari ST brechen goldene Zeiten für Basic-Programmierer an

Preiswerter AT-Kompatibler 90
Mit der unhandlichen Bezeichnung AT4SLC kommt ein attraktiver AT-Kompatibler daher

Januskopf 92
Apple versucht sich mit dem Apple-IIGS, einem Rechner ohne besondere technische Raffinessen, auf dem Markt zu behaupten

Harddisk auf Steckkarte 98
Einfacher geht es nicht: 32-MByte-Festplatte auf einer Steckkarte

mc-hard

Der EMUF08 101
Einplatinencomputer mit der 68008 für anspruchsvolle Aufgaben zum Selberbauen

Festplatten-Controller für den PC 108
Zwei neue Controller von Western Digital

mc-leserbefragung 109

mc-markt 112

mc-vorschau 142

Impressum 140

Thema Btx

Die gut geschriebenen Artikel zur „Kommunikation" in der mc-Ausgabe 12/86 passen nicht ganz zu der reißerischen Btx-Überschrift. Das Medium Btx ist etabliert und es bedarf nur beharrlicher Überzeugungsarbeit, um auch weiterhin ein kontinuierliches Wachstum der Nutzerzahlen sicherzustellen. Bringt man den privaten Haushalten, also den eigentlich anvisierten Endnutzern die Vorteile und Möglichkeiten ausreichend nahe (von Btx-Kontoführung über Btx-Telefonbuch und Btx-DB-Fahrplänen bis zu Btx-Reisebuchungen und vieles mehr), so dürfte eine überraschende Akzeptanz erreicht werden. Aus meiner Erfahrung mit der Beratung bezüglich Btx ist regelmäßig ein hoher Akzeptanzerfolg zu verzeichnen, wenn der Nutzer selbst unter Anleitung mit Btx arbeitet. Hilfreich ist es dann nicht, wenn in Ihrem Artikel die Post angeblich nicht an den Einzug in private Haushalte glaubt. Das ist schlichtweg falsch. Es wäre besser, die journalistischen Energien darauf zu verwenden, das neue Medium zu nutzen und an der Verbesserung mitzuarbeiten, um so den mc-Leser auch von Btx profitieren zu lassen.

Dipl.-Ing. Klaus Hänsel,
6101 Fränk.-Crumbach

mc-68-ECB-Platine ab sofort lieferbar

Der Beitrag über die Ausrüstung eines Z80-Systems mit einer 68000-CPU, der in der mc 1/87 seinen Anfang hatte und in dieser Ausgabe fortgesetzt wird, ist auf ein unerwartet großes Interesse gestoßen. Der Autor des Artikels, Stefan Wurdack, hat sich deshalb bereit erklärt, die Platine mit dem Monitor-EPROM und einem kleinen Handbuch direkt an die Interessenten zu versenden. Die Platine kann mit den Betriebssystemen OS-9 und CP/M-68k betrieben werden. Die Anpassung an CP/M-68k wird in der nächsten Ausgabe beschrieben.
Wer also sein 8-Bit-CP/M-System zu einem vollwertigen 16-Bit-System ausbauen möchte, kann sich direkt an den Autor wenden:

Stefan Wurdack,
Alte-Post-Straße 1f,
8011 Baldham

Rekursion in Turbo-Pascal

Hört sich gut an, was in dem Artikel in Ausgabe 11/86 versprochen wird. Ich bin ganz sicher, daß ich die Änderungen in meiner Turbo-Pascalversion (3.00A) richtig vorgenommen habe. Doch leider wird bei meinem Rechner (CPC-6128) nach Aufruf des Programmes nach Bild 1 aus dem Beitrag in den Bildschirmspeicher geschrieben und der Rechner bleibt hoffnungslos hängen. Auf welchem Rechner läuft der Patch?

Ralf Krause,
5000 Köln

Antwort

Bei der Installation des Patches ist darauf zu achten, daß der verwendete Debugger auch bei sich überlappenden Speicherbereichen den Verschiebebefehl korrekt ausführt. Der erwähnte DDT ist dazu nicht in der Lage, wohl aber zum Beispiel Trace-80. Anstelle eines Verschiebebefehls kann man jedoch auch zwei Verschiebebefehle benutzen: 3.00A : Verschiebe 04D9..04EB nach 78D9
Verschiebe 78D9..78EB nach 04DC
3.00A: Verschiebe 0509..051B nach 7A09
Verschiebe 7A09..7A1B nach 050C
Diese Verschiebebefehle führt auch DDT richtig aus. Noch ein weiterer Hinweis: Die Schachteltiefe der Rekursion darf nicht zu groß werden, denn sonst läuft der CPU-Stack in den Rekursionsstack hinein und zerstört dort Daten. Im Turbo-Pascalhandbuch ist das geschildert und auch, wie man die Schachteltiefe erhöhen kann.

Martin Weiß,
Stefan Fuchs,
7500 Karlsruhe

Disketten-Speedy

Aufgrund mehrerer Anfragen hier einige grundsätzliche Bemerkungen:

Das Assemblerlisting für den Disketten-Speedy wurde von mir auf dem Microsoft-Makroassembler 4.0 entwickelt und wird in der veröffentlichten Form von diesem fehlerfrei assembliert. Wie ich inzwischen von vielen Seiten erfahren habe, sind die älteren Versionen des Microsoftassemblers leider nicht in der Lage, alle Assemblerbefehle in allen erlaubten Adressierungsarten zu verstehen. Der sehr leistungsfähige Assemblerbefehl LES (Lade Zeiger über ES-Register) gehört wohl zu den Befehlen, die erst ab Version 4.0 fehlerfrei verarbeitet werden. Mancher Leser, dessen Assembler nicht mitspielte, hat mir „Ersatz" in umständlicher Codierung zugesandt, was sicherlich ein Weg aus diesem Dilemma ist. Zur Schande von Microsoft muß jedoch aufgezeigt werden, daß das gute alte DEBUG.COM, welches wie bekannt zum MS-DOS-System gehört, den von mir geschriebenen Code richtig versteht. Gibt man Programme mit dem E-Befehl ein, so hat man das Programm in wenigen Minuten in einer lauffähigen Form. Will man aber den A-Befehl benutzen, so muß man die DEBUG-spezifischen Assemblerbefehle nehmen. Um dieses Kompatibilitätsproblem der verschiedenen Assembler zu verdeutlichen, sollte man vielleicht bei jeder Veröffentlichung mit angeben, unter welcher Assemblerversion der jeweilige Programmtext lauffähig ist. Einen Quelltext zu schreiben, der auf allen auf dem Markt befindlichen Assemblern lauffähig ist, wird wohl immer weniger möglich sein.

Gerhard Runge,
4200 Oberhausen

Anmerkung der Redaktion: Die Anregung aus dem Schlußsatz des Briefes von Herrn Runge wollen wir gern aufgreifen. Dazu bitten wir alle, die ein Listing einsenden, den Kopf ohne Schnick-Schnack, aber mit der Assemblermeldung zu gestalten.

Buchbesprechung Btx-Report

Als interessierte und regelmäßige Leserin Ihrer immer informativ-kritischen Computer-Zeitschrift möchte ich mir heute doch erlauben, eine kleine Kritik anzubringen: Sie behaupten in Ihrer Rezension, daß die meisten der (im Report enthaltenen Btx-Systembeschreibungen) Informationen beim Händler oder bei der Bundespost selbst auf einfachem Weg zu beziehen sind. Hiervon leiten Sie ab, unser Preis von 80 DM liege zu hoch. Probieren Sie doch einfach mal selber aus, vergleichende Daten von der Post, Händlern oder Herstellern zu erhalten. Nicht von ungefähr wird unser Progris-Btx-Report sowohl von der Bundespost, der Btx-Anbietervereinigung als auch von vielen Fachhändlern bezogen und als aktueller Status-Quo geschätzt. Kein Wunder: Immerhin werden von uns alle Technik-Informationen alljährlich brandaktuell mit viel Akribie und durch wochenlange eigene Recherchen erhoben. Im übrigen betrachtet die Firma Progris den Btx-Report in erster Linie als einen flankierenden Service zu ihren allgemeinen Btx-Dienstleistungen. Bisher läßt sich mit der gedruckten Marktübersicht kein Geschäft machen: nicht einmal die Herstellungskosten sind gedeckt!

Progris GmbH,
Heike Liß,
1000 Berlin 33

Dreimal Schaltungspraxis, einmal Turbo-Pascal

Mehrere neue Sonderhefte sind jetzt vom Franzis-Verlag erhältlich. Drei davon setzen die Reihe „Schaltungspraxis" fort, die aus der gleichnamigen Rubrik der Zeitschrift ELEKTRONIK entstanden ist. Sie befassen sich mit den Themen
– Analog-, Operationsverstärker- und Stromversorgungsschaltungen,
– Oszillatoren, Generatoren, Rechen- und Filterschaltungen,
– Steuer- und Regelschaltungen.

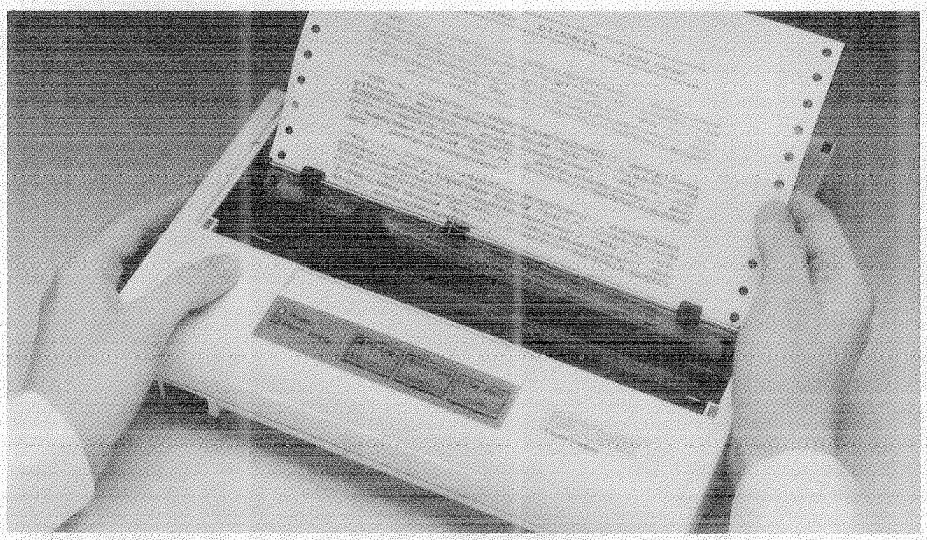
Ein weiteres Sonderheft wendet sich an Programmierer und Software-Freaks. Es faßt die bisher in der mc erschienenen Beiträge zum Thema Turbo-Pascal zusammen. Viele der abgedruckten Programme sind auch auf Sammeldisketten zu haben. Wie die drei Schaltungspraxis-Sonderhefte kostet auch das Turbo-Pascal-Sonderheft 19,50 DM und ist über den qualifizierten Buch- und Zeitschriftenhandel oder direkt beim Verlag zu beziehen.

Einbanddecken für Ihre mc-Sammlung

Wer seine wertvolle mc-Sammlung nicht als lose Einzelhefte, sondern als gebundene Bücher aufbewahren möchte, kann Einbanddecken aus Graupappe mit Ganzleinenüberzug verwenden. Die Einbanddecken von der Buchbinderei Neumann aus Ober-Warngau kosten 11,50 DM und bieten genügend Platz für den Jahrgang 1986. Die Einbandarbeiten kann jeder Buchbinder durchführen.

Handlicher Drucker für handliche Computer

Das Angebot an tragbaren Computern ist reichhaltig, aber leider halten die Peripheriegeräte damit nicht ganz mit. Daß man auch kleine Drucker mit großer Leistung herstellen kann, hat jetzt Diconix bewiesen. Deren Tintenstrahlmodell 150 wiegt weniger als 2 kg und mißt knapp 28 cm * 17 cm * 5 cm. Trotzdem erreicht es eine Geschwindigkeit von 150 Zeichen/s mit Beinahe-Briefqualität. Das Gerät wird von Akkus oder einem Netzadapter aus mit Spannung versorgt. In der Standardversion, die etwa 1500 DM kostet, ist es mit Centronics-Schnittstelle ausgestattet. Ein serielles Interface ist ebenfalls erhältlich. Der Diconix-150 beherrscht verschiedene nationale Zeichensätze und Schriftarten (auch Proportionalschrift). Er bedruckt das Papier bei Bedarf auch vertikal. Das ist z. B. bei „Spreadsheets" sehr nützlich, wenn die normale Papierbreite nicht ausreicht. Die Tinte befindet sich in einer Patrone, die leicht auszuwechseln ist. Verschiedene Farben sind erhältlich. Der Inhalt einer Patrone reicht für etwa 500 Seiten Text. Neben Normalpapier lassen sich auch Folien, Umschläge und – für beste Qualität – spezielles Ink-Jet-Papier bedrucken.



Diconix, eine Tochterfirma des Eastman-Kodak-Konzerns, hat diesen handlichen Tintenstrahldrucker entwickelt

mc-Programme auf Diskette

Der Franzis-Software-Service bietet wieder eine interessante Sammeldiskette für MS-DOS-Computer an. Auf der MS-DOS-Sammeldisk 6 sind interessante Programme aus den mc-Ausgaben 10/86 und 11/86 zusammengefaßt und teilweise verbessert worden. Ein Schwerpunkt liegt beim Einsatz von Grafik. So wurden zum Beispiel die Programme Fractal und DGD (Entwicklung von Drucker-Grafiken), aber auch Baum (grafische Anzeige der Datei-Hierarchie) und Bigchr (Druck von Großbuchstaben) auf die Diskette übernommen. Zusätzlich ist das Jahresinhaltsverzeichnis der mc/86 auf der MS-DOS-Sammeldisk 6 zu finden, die beim Franzis-Software-Service für 20 DM zu haben ist. Für den gleichen Preis gibt es die CP/M-Sammeldisk 6, die einige Programme aus der mc 12/86 enthält. Auf dieser Diskette sind enthalten: Backup, USE, Ordnung auf Diskette und RAM-Floppy-Initialisierung.

Befehle für 1-2-3 in Klartext

Unter der Bezeichnung HAL (Human Access Language) bietet Lotus jetzt ein speicherresidentes Begleitprodukt zu Lotus 1-2-3 an, mit dem Befehle nicht mehr aus einem Menü ausgewählt, sondern in Klartext formuliert werden. Mit dem bislang nur in englischer Sprache arbeitenden Programm können zusätzliche Funktionen aufgerufen werden. So können zum Beispiel Arbeitsblätter verknüpft oder Beziehungen von Feldern angezeigt werden.

Die Eingabe von Formeln, das Kopieren von Daten und die Sortierung von Datenbanken wird durch HAL vereinfacht. Das 380 DM teure Programm soll auch ins Deutsche übersetzt werden.

ATARI 1040 ST. Spitzentechnologie, um mehr zu leisten.

Der ATARI 1040 ST hat alle Merkmale, die Sie heute von einem 16/32-bit Computer erwarten können – sei es die Speicherkapazität, die hohe Arbeitsgeschwindigkeit, die bestechende Grafik, die Schnittstellen...

In dieser Leistungsklasse hat ATARI die Maßstäbe gesetzt. Auch beim Preis!

Und Computerleistung zu solch niedrigen Preisen kann Ihnen nur bieten, wer modernste Technologie einsetzt.

ATARI. Das ist Computertechnologie von heute für Menschen, die mit mehr Leistung mehr leisten wollen.

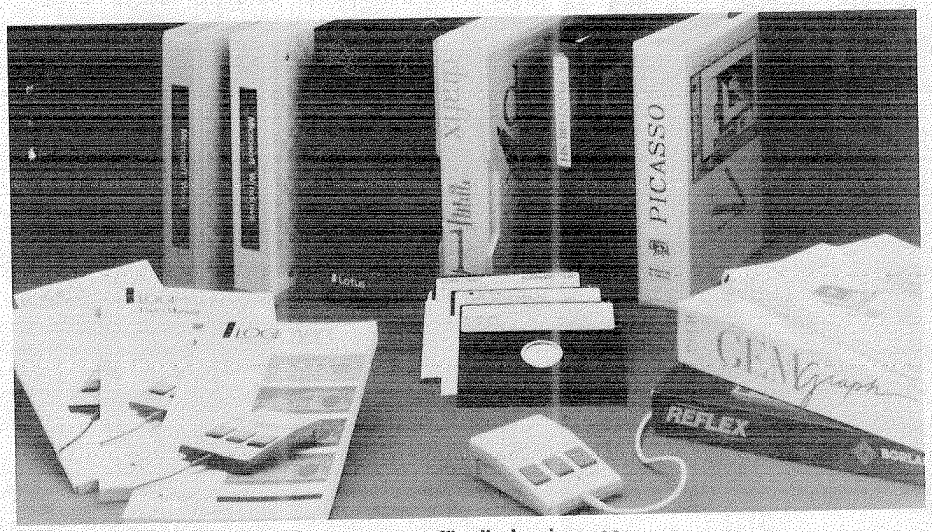
ATARI 1040 ST – bei Ihrem Fachhändler.

ATARI

...wir machen Spitzentechnologie preiswert.

Attraktive Maus-Kombinationen

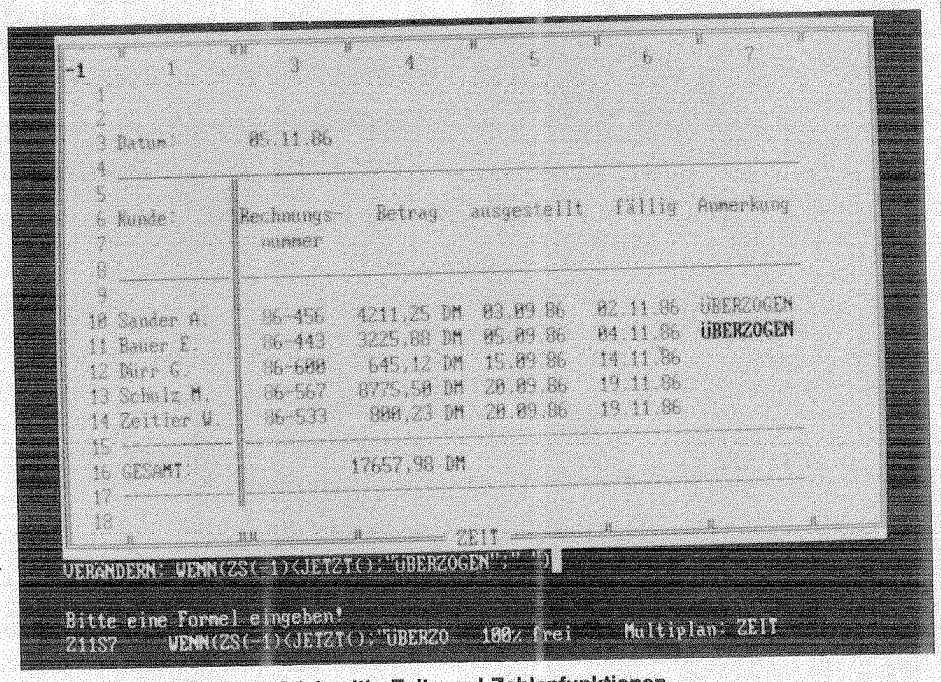
Die schweizerische Firma Logitech S.A. bietet ihre Logimouse-C7 jetzt in Kombinationen mit verschiedenen Softwareprodukten an. So gehören zum sogenannten „Plus Package" die „Point-and-Click-Shell" für Lotus 1-2-3, das interaktive Menüsystem „Logimenu", das speicherresidente Maus-Konfigurationsprogramm „Click" und der Texteditor „Point". Zur CAD-Station wird der PC mit dem Paket „LogiCADD". Es enthält unter anderem ein Programm, mit dem man Zeichnungen auf einem Matrixdrucker ausgeben kann. Auch Borlands Datenbanksystem „Reflex" gibt es als Paket zusammen mit der Logimouse. Ein weiteres Programm erlaubt es, technische Zeichnungen mit Freihandzeichnungen zu mischen.



Eine reiche Auswahl an Software gibt es für die Logimouse

Multiplan 3.0 in Deutsch

Multiplan, das Tabellen-Kalkulationsprogramm von Microsoft, das Mitte 1982 auf den Markt kam und von dem mehr als 1,4 Mio. Kopien verkauft wurden, ist in der Version 3.0 um rund 30 neue Funktionen gegenüber Version 2.0 erweitert worden. Es ist sofort in der deutschen Version lieferbar. Bei Multiplan 3.0 ist eine Zeitmanagement-Funktion integriert, mit der Termine und Abläufe verfolgt und protokolliert werden können. In sämtlichen Rechnungen lassen sich Zeit und Datum berücksichtigen. Durch die unterschiedlichen Kontrollfunktionen der Version 3.0 behält der Benutzer im Arbeitsblatt immer den Überblick, denn Formelzusammenhänge lassen sich jederzeit anzeigen und Verknüpfungen bis zu ihrem Ursprung zurückverfolgen. Fehler lassen sich so wesentlich leichter finden, und der Aufbau einer Tabelle ist einfacher nachzuvollziehen. Mit der Version 3.0 ist es erstmals möglich ASCII-Daten zu verarbeiten. Damit hat der Anwender eine neue Möglichkeit zum Datenaustausch auf der Basis eines Zeichenstandards. Als erstes Tabellenkalkulationsprogramm der Welt ist dieses zusätzlich in einer Netzwerkversion erhältlich.



Multiplan 3.0 bearbeitet gleichzeitig Zeit- und Zahlenfunktionen

Integral AT

In der Marktübersicht über PC-artige Computer in der mc 11/86 haben wir einen Computer von Hewlett-Packard mit der Bezeichnung „HP-Integral-PC" ausgewiesen. Jetzt wurden wir von der Firma Integral aus 4173 Kerken darauf hingewiesen, daß „diese Bezeichnung eine Verletzung unserer (Integral-)Rechte" darstellt. Unter der Bezeichnung Integral AT 286/8 wird von der Integral International GmbH ein Personal Computer montiert und vermarktet. Der Integral AT basiert auf einem ALR-Motherboard mit der Intel-80286-CPU, die mit 8 MHz getaktet wird. Zu einem Preis von etwa 8550 DM erhält man den Integral AT mit 512 KByte RAM, einer 20-MByte-Festplatte, einem Floppy-Laufwerk (1,2-MByte- und 360-KByte-Format), MS-DOS 3.0, 3.1, Xenix und einem Netzwerk. Die Integral International GmbH, die wir um Entschuldigung für die „Verletzung ihrer Rechte" bitten, bietet Erweiterungskarten für den Integral AT an.

Der Integral AT

Euromicro-Weltsymposium

Vom 14. bis zum 17. September 1987 findet in Portsmouth, Südengland, das 13. Euromicro-Weltsymposium statt. Dieser Weltkongreß ist das führende europäische Forum für Mikroelektronik, Mikroprozessor-Systeme und Mikrocomputer. Ein Hauptanliegen ist der Erfahrungsaustausch und die Weitergabe von neuesten Entwicklungen zwischen Universitäten und Industrie. Verantwortlich für den wissenschaftlichen Inhalt der Konferenz ist Dr. Harald Schumny, Postfach 3345, 3300 Braunschweig. Der Hauptteil der Veranstaltung wird aus etwa 150 ausgewählten Fachvorträgen bestehen.

Kopieren erlaubt

Shareware, eine Software-Verkaufsstrategie, die sich in den USA schon größter Beliebtheit erfreut, hat nun auch bei uns Fuß gefaßt. Und das funktioniert so: Dem Käufer ist es erlaubt, ein Programm beliebig zu kopieren und an Freunde und Bekannte weiterzugeben. Wer das Programm voll nutzen möchte, kann beim Hersteller das Handbuch erwerben. In diesem Fall erhält er auch weitere Unterstützung, z. B. neue Versionen.
Die Münchner Firma pc-plus praktiziert dieses Konzept jetzt mit ihrem Menüsteuerprogramm pc+master, das über ausgewählte Händler an den Mann gebracht wird. Die Idee ist sicher nicht schlecht: Statt einen hoffnungslosen Kampf gegen Software-Piraten zu führen, legalisiert man, was ohnehin nicht zu unterbinden ist, und versucht, dennoch ein Geschäft damit zu machen.

Pannendienst

Einen Notdienst bei Problemen mit Standard-Software und mit der Hardware können PC-Anwender jetzt in Anspruch nehmen. Die Firma Raymund Witte adiuvans unterhält in Köln eine Hot-line, die Montag bis Freitag zwischen 9.00 und 18.30 Uhr besetzt ist. Sie garantiert innerhalb von vier Stunden einen Rückruf oder innerhalb eines Tages den Service vor Ort. Voraussetzung für diese Hilfsbereitschaft ist ein Vertrag, der im allgemeinen auch die einführende Schulung auf dem PC enthält. Die Pannenhelfer kennen sich mit Siemens- und Nixdorf-Rechnern ebenso wie mit Olivetti- und Commodore-Systemen aus.

Power Supply '87

Vom 24. bis zum 27. Juni 1987 findet in Frankfurt die Messe Power Supply zum ersten Mal statt. In der Bundesrepublik wurden im letzten Jahr über 2,4 Milliarden DM für Stromversorgungen ausgegeben. Das ist sicher Grund genug, sich über den Stand der Technik zu orientieren, die für gleichbleibende Spannung auch in jedem Computer sorgt. Da auf den zahlreichen Messen in Deutschland das Marktsegment der Stromversorgungen bislang nicht genügend zur Geltung kam, treffen sich in Frankfurt Händler, Bauteileherstellen und Zubehöranbieter, die dem Besucher alle gewünschten Informationen über gesicherte Stromversorgung geben wollen. Zum Start der neuen Messe werden etwa 100 Aussteller und bis zu 5000 Fachbesucher erwartet.

OSIS XT-Profi-System
IBM-KOMPATIBEL
Mainboard mit 256 KB RAM bestückt,
auf Mainboard bis 640 KB RAM bestückbar,
CPU 8088 (optional 8087),
Color-Grafik-Karte oder
Monochrome-Grafik-/Printer-Karte,
Multi-I/O-Karte mit Gameport,
akkugepufferte Uhr mit Kalender,
serielle und parallele Schnittstelle,
1 × 360 MB, Disk-Drive
stabiles Klappdeckelgehäuse,
135-W-Netzteil,
Tastatur mit deutscher Norm
Komplett 1189.–

OSIS AT-Profi-System
IBM-KOMPATIBEL
schnelle 10-MHz-Version
Mainboard mit 640 KB RAM bestückt,
auf Mainboard bis 1 MB RAM bestückbar,
CPU 80286 (optional 80287),
Color-Grafik-Karte oder
Monochrome-Grafik-/Printer-Karte,
akkugepufferter Setup, Uhr, Kalender,
Taktrate 10/8/6 MHz umschaltbar,
FDD-Karte (Disk-Controller),
Disk-Drive 1,2 MB/360 KB umschaltbar,
Tastatur mit deutscher Norm,
Stahlblechgehäuse und 200-W-Netzteil,
ausführliches Handbuch
Komplett 2689.–

IBM-XT-/AT-kompatibel

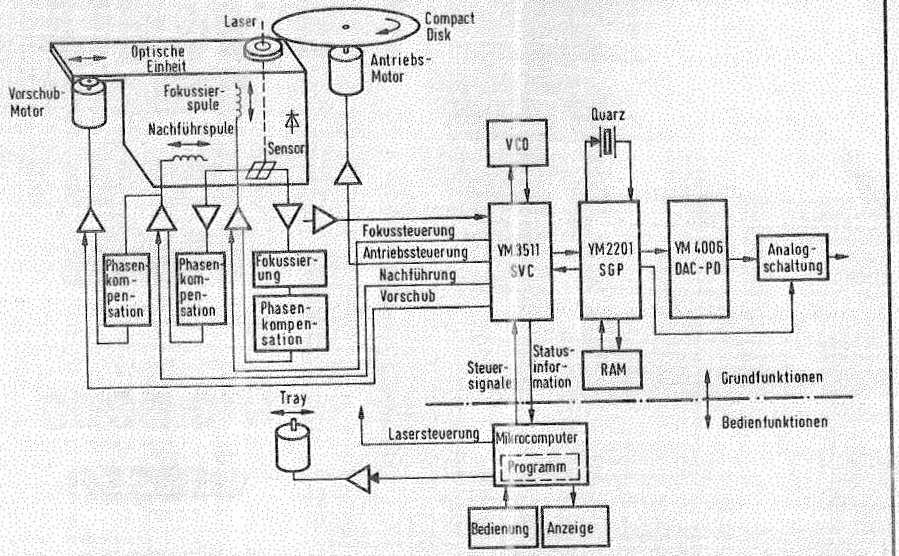
Erweiterungen für OSIS-XT-Profi-System	
– Speicherausbau auf 640 KB	DM 140.–
– zweites Disk-Drive, 360 KB	DM 298.–
– 20-MB-Festplatte mit Controller	DM 1299.–
– Aufpreis für 150-W-Netzteil	DM 39.–
Erweiterungen für OSIS-AT-Profi-System	
– zweites Disk-Drive, 1,2 MB	DM 398.–
– 20-MB-Festplatte mit Controller	DM 1579.–
– serielle/parallele Karte	DM 165.–
XT-Komponenten:	
Mainboard, 640 KB RAM (0 KB best.)	DM 369.–
Turbo-Board, 8 MHz (0 KB best.)	DM 384.–
EGA-Karte mit Hercules-Emulation (voll IBM-EGA-kompatibel)	DM 748.–
Color-Grafik-Karte	DM 149.–
Monochrome-Grafik-/Printer-Karte (Hercules-komp.)	DM 196.–
Multi-I/O (mit Disk-Controller)	DM 220.–
I/O-Plus-Karte (ohne Disk-Controller)	DM 186.–
Floppy-Disk-Controller (2 Drives)	DM 85.–
Floppy-Disk-Controller für 1,2 MB Drive on XT	DM 276.–
Serielle RS232-Karte	DM 78.–
512-KB-RAM-Karte (langer Steckplatz, 0 KB best.)	DM 120.–
576-KB-RAM-Karte (kurzer Steckplatz, 0 KB best.)	DM 120.–
Parallele Printer-Karte	DM 58.–
Tastatur mit deutscher Norm	DM 198.–
Schaltnetzteil, 135 W, mit seitlichem Schalter	DM 199.–
Stabiles Stahlblechgehäuse mit Klappdeckel	DM 145.–

Fordern Sie unser kostenloses AT-System-INFO an!

IBM und HERCULES sind eingetragene Warenzeichen. Auf alle Produkte 6 Monate Garantie. Technische Änderungen vorbehalten. Zwischenverkauf vorbehalten. Versand per Nachnahme unfrei. Lieferbedingungen auf Anfrage.

Versand:
Telefon 0 23 03/2 23 11, Postfach 19 63, 4750 Unna
Oliver Schlüter · Informations-Systeme – Öffnungszeiten: Mo.–Fr. 9–13 und 14–18 Uhr

Ladenverkauf:
Gerh.-Hauptm.-Str. 20, 4750 Unna, Tel. 0 23 03/1 24 36, Jürgen Heitmann – Elektronik



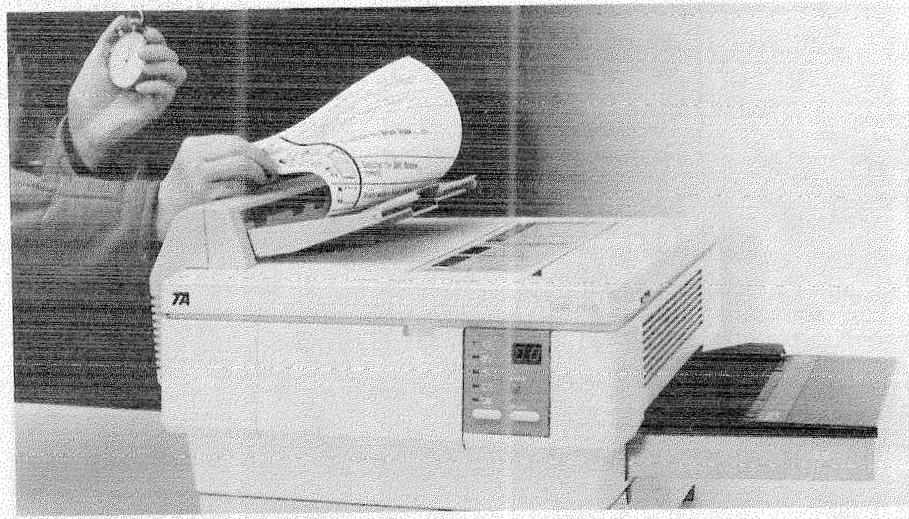
Der schematische Aufbau eines CD-Players, der die neuen IC von Yamaha einsetzt

Schaltkreise für CD-Player

Speziell für die Anwendung eines CD-Players als Massenspeicher hat Yamaha einen Schaltkreis für die Umwandlung der Signale entwickelt. Neben den beiden IC YM-3511 und YM-2201, die für Demodulation der EFM-Signale, die Systemkontrolle und die Fehlerkorrektur sorgen, wandelt der 16-Bit-D/A-Wandler YM-4006 die seriellen Digital-Daten in analoge Daten um. In diesen Baustein ist ein Entkoppelverstärker integriert, der in 2,7 µs auf seinen stationären Endwert einschwingt. Der bisher notwendige Aufwand an Mikroprozessoren und entsprechender Peripherie wird durch die IC von Yamaha auf ein Minimum reduziert.

In sechs Sekunden eine DIN-A4-Seite

Mit einer Geschwindigkeit von sechs Sekunden für eine DIN-A4-Seite arbeitet der neue Laserdrucker SDR7710 von Triumph-Adler. Für Hobbyzwecke ist das Gerät allerdings noch etwas zu teuer, denn es soll knapp 10 000 DM kosten. Wer auf viele verschiedene Schriftarten Wert legt, ist mit diesem Modell gut bedient. Der SDR7710 kann mit knapp 40 Schriftarten aufwarten, wovon sich drei dynamisch vergrößern und verkleinern, dehnen oder neigen lassen.



Mit einem Halbleiterlaser arbeitet dieser Drucker von Triumph-Adler

Zahlreiche Strichcodes beherrscht das Gerät ebenfalls. Die Anpassung an unterschiedliche Programme bereitet keine Schwierigkeiten, da der SDR7710 sieben weitverbreitete Matrix-, Typenrad- und Laserdrucker emuliert. Mit 29 kg und einer Standfläche von 45 cm × 45 cm paßt der Drucker in jedes Büro. Und damit er dort nicht unangenehm auffällt, ist er sehr leise (47 dBA).

10 MByte für Schneider-Joyce

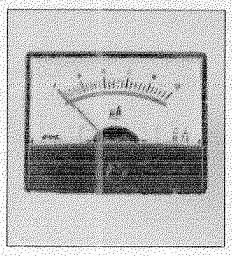
Für nur 1800 DM kann man jetzt das Textsystem Joyce um eine 10-MByte-Festplatte erweitern. Dawes Electronics aus München bietet das Erweiterungskit mit einem erweiterten CP/M plus an. Der Anschluß erfolgt über den Expansionport des Joyce, der sich an der Rückseite des Monitors befindet. Der Erweiterungsstekker ist durch die Zusatzbox geführt und kann für andere Anwendungen weiter benutzt werden. Das Gehäuse enthält neben dem 10-MByte-Laufwerk den Controller und ein eigenes Netzteil.

10 MByte am Schneider-Joyce für professionelle Anwendungen

Commodore-Interface erlaubt Druckeranschluß

Ein neuentwickeltes Interface von Citizen ermöglicht bei allen Commodore 64, 128 und VC-20 den Anschluß eines 120D-Druckers von Citizen. Dabei können internationale Zeichensätze durch verschiedene DIP-Schalter oder Software-Anweisungen ausgewählt werden. Die zwei 6-Pol-DIN-Verbindungen machen es auch für ein Commodore Daisy-chain-Netzwerk-Interface benutzbar. Das Interface kostet rund 200 DM und wird mit zweijähriger Garantie verkauft.

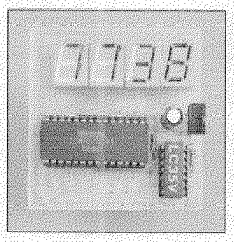
Drehspul-Einbauinstrumente:



Skala 60×45 mm, Einbauflansch 38 mm Ø, Tiefe 32 mm. Klasse: 2,5, moderne rechteckige Form, Abdeckung industriegrau, Flachbogenskala, Nullpunktkorrektur, alle Instrumente sind für Beleuchtung vorbereitet und verdrahtet. Mit 2 Lampen à 6 V (Best.-Nr. 0503529) wird die Skala indirekt beleuchtet.

EW-60: **Stück 22,50 DM**

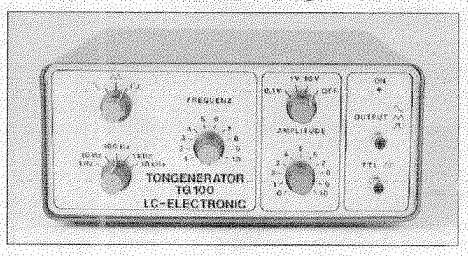
Best.-Nr.	Typ	Ri.
0602994	50 µA	1,2 kOhm
0600674	± 50 µA	550 Ohm
0600683	100 µA	550 Ohm
0600692	1 mA	150 Ohm
0600709	500 mA	1 Ohm
0600718	1 A <	1 Ohm
0600727	3 A <	1 Ohm
0600736	5 A <	1 Ohm
0600745	10 A <	1 Ohm
0600754	15 V	15 kOhm
0600763	25 V	25 kOhm
0600772	30 V	30 kOhm
0600781	50 V	50 kOhm
0600790	300 V	300 kOhm



RENKFORCE®-Vor-/Rückwärtszähler:

Elektronischer Zähler mit rotleuchtender 13-mm-LED-Anzeige, die wahlweise senk- oder waagerecht eingebaut werden kann. – Besonders einfacher und problemloser Aufbau durch **IC-Technik (ICM-7217).** Betriebsspannng.: 7–12 V= (max. 200 mA). Nullsetzung per Taster, exakte **Impulszählung von 1–9999** durch elektron. Kontakt oder präzisen (prellfreien) Schalter. Ideal als Basis-Baustein für digitale Bandzählwerke, wobei das setzbare Register auch NULLSTOP o. ä. Funktionen erlaubt. Personenzählung mit entspr. Lichtschranke u.v.a. mehr. Kompl. mit allen Bauteilen, gedr. Platine und Anleitung (o. Netzteil).
Best.-Nr. 0400970 ..... **98,50 DM**

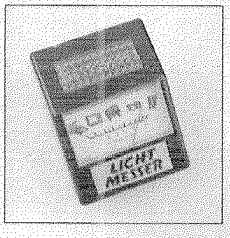
Für Hobby, Service und Schulung!



Funktionsgenerator 1 Hz–100 kHz:

Dekadische Bereichsumschaltung, lineare Skala und ergonomische Anordnung der Bedienelemente. Ausgang (symmetrisch) umschaltbar auf Sinus, Dreieck, Rechteck und TTL, Lautsprecher (8 Ohm), direkt anschließbare Bereiche: 1–10 / 10–100 / 100–1000 Hz/ 1–10 / 10–100 kHz, Ausgang 0–0,1 / 1,0 / 10 Volt, Cynch-Buchsen, dunkelbraun lackiertes Stahlblechgehäuse mit Alufront, B×H×T: 200×100×220 mm, dreiadrige Netzleitung 1,5 m, Lieferung mit Anleitung und Schaltplan.
Best.-Nr. 0603136 ..... **279,– DM**

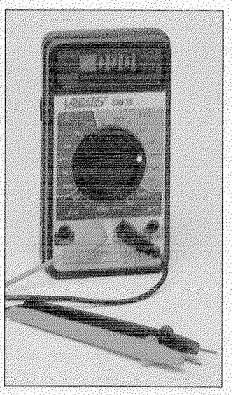
Wichtig für Ihre Gesundheit.
Richtige Beleuchtung schont die Augen.



Lichtmeßgerät „LUX-3":

Immer sekundenschnelle Messungen. Für Büro, Haushalt, Arbeitsplatz, Hobby, Sport, Schule u.v.a.m. Handliches Gerät komplett mit Pflanzenlichttabelle.
Best.-Nr.
0603993 ..... **17,95 DM**

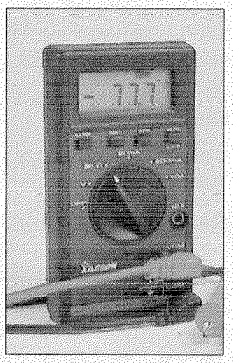
Preisschlager ***Praxisfreundlich mit 10-Amp.-Bereich!***



LCD-Digital-Multimeter LDM-7 S:

3½stelliges, robustes Multimeter **mit praktischem Drehschalter.** Überlastschutz (2 A 5×20 Sicherung) in allen Bereichen außer 10 A=. Zusätzl. Dioden-Test. Versenkte 4-mm-Buchsen, 10 mm LCD-Anzeige. **Eingangswiderstand 10 MOhm.** Genauigkeit: ± 1 Digit.
Daten:
V=: 0–200 mV/2/20/200/1000 V
V~: 200/750 V
A=: 0–200 µA/2/20/200 mA/2/10 A
Ohm: 0–200 Ohm/2/20/200 kΩ/2/20 MOhm.
B×H×T: 80×150×33 mm, mit Meßkabeln, Ersatz-Sicherung und Anleitung (engl.).
Best.-Nr. 0603877 ..... **69,50 DM**
Passende 9-V-Batterie „UM-5"
Best.-Nr. 0311530 ..... **1,50 DM**

- ***Ein Preisschlager in kompakter Ausführung (nicht größer als eine Zigarettenschachtel).***
- ***Vollgepackt mit modernster Technik, autom. Bereichswahl, Datahold, Memoryfunktionen und Durchgangsprüfer.***
- ***Automatische Polaritätsanzeige.***
- ***Praxisgerecht (in der Sicherheitsfarbe „gelb"!) und robust.***

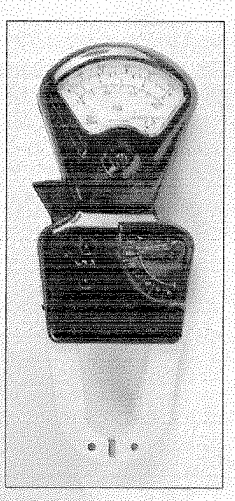


10-Amp.-LCD-Digital-Multimeter „KD-568 Yellow": Neu!

Qualitätsgerät für den Einsatz in Schulen, Labor, Service, Industrie, Kfz-Elektronik und Hobby. Sehr gut lesbare 3½stell. LCD-Anzeige (10 mm) mit Meßbereichsanzeige auf dem Display. Übersichtliche Bedienelemente mit praktischem Drehschalter. Hohe Betriebssicherheit, voller Bereichsschutz und Sicherheits-Prüfschnüre. Max. Anzeige 1999. Überbereichsanzeige, Batteriezustandsanzeige, eingeb. Pieper für Durchgangstest. Versenkte 4-mm-Buchsen. Eingangswiderstand 100 MΩ im Bereich 200 mV, alle anderen Bereiche 10 MΩ. Überlastschutz DC 1000 V, AC 750 V. B×H×T: 69×127×25 mm. Inkl. Meßkabeln, Batterie 9 V und engl. Anleitung. **Lieferung in Gelb.**

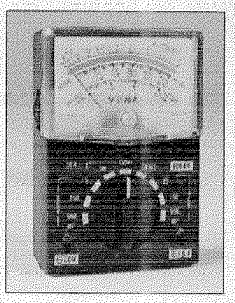
Daten:
V=: 200 mV/2/20/200/1000 V ± 0,8 %
V~: 2/20/200/500 V ± 1,2 %
A=: 20 mA/200 mA/ 10 A ± 1,2 %
A~: 20 mA/200 mA/ 10 A ± 1,2 %
Ohm: 200 Ω/2/20/200 kΩ/2 MΩ

Best.-Nr. 0605303 ..... **89,– DM**



Stromanleger-Zange U-91:

Kontrolle von Strömen und Spannungen in Installationsanlagen. Einhandbedienung, Umschaltung von Strom- auf Spannungsmessung ohne Auftrennen des Stromkreises.
Wechselstrom:
0–10/25/100/250/500 A ± 4 %
Wechselspannung:
0–300/500 V_{eff} ± 4 %
Abmessungen/Gewicht: B×H×T: 94×238×36 mm/ca. 0,6 kg
Zubehör:
Prüfkabelsatz, Tragetasche, Bedienungsanleitung.
Best.-Nr.
0601888 . **nur 79,50 DM**

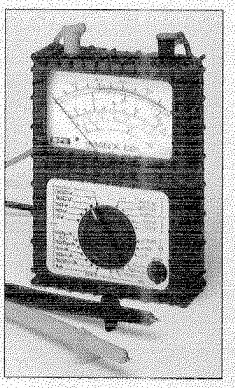


Preisschlager

Mini-Multimeter-9120:

Preiswertes Modell mit den gebräuchlichsten Bereichen, nicht nur für den Einstieg in die Elektronik. Besonders günstige Abmessungen, überall und sofort einsatzbereit (B×H×T: 63×90×33 mm).

Daten: **V=:** 0–15/150/500/1000 V (1 kΩ/V)
V~: 0–15/150/500/1000 V (1 kΩ/V)
A=: 1 mA–150 mA
Ohm: 0–100 kΩ
Inkl. Prüfschnüre, Batterie und Anleitung.
Best.-Nr. 0603966 ..... **19,80 DM**

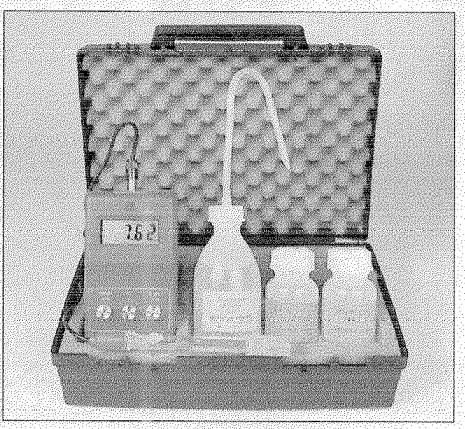


Analog-Vielfachmeßgerät „MAN-X 02":

Das MAN-X 02 ist leicht und nicht größer als Ihre Brieftasche. Sehr übersichtliche Einhandbedienung über Drehschalter aller 23 Meßbereiche durch farbige Skalen. Aufwendiger Überlastschutz mit 1 und 10 Amp. Hochleistungssicherungen. Anzeige einer anstehenden Spannung >80 V durch Signallampe bei defekter Sicherung. Stromversorgung 1,5 V Mignon (1 Batt. inkl.). B×H×T: 102×145×42 mm. Mit Sicherheitsmeßkabeln und Ersatzsicherungen. 20000 Ω/V=

V=: 0–0,1/3/10/30/100/300/1000 V
V~: 0–10/30/100/300/750 V
A=: 0–50/100µA/1/10/100 mA/1/10 A
A~: 0–1/10/100 mA/1/10 A
Ohm: 0–5/50/500Ω/10/100 kΩ/1MΩ
Best.-Nr. 0604796 ..... **199,50 DM**

Im praktischen Transportkoffer!



Elektronisches P_H-Meter:

Fertiggerät mit hochwertiger Elektrode. Ein Präz.-Meßgerät für die Bereiche: Umweltschutz, Labortechnik, Landwirtschaft, Blumenzucht, für den Aquarianer u. a. pH: 0–14.00, Tol.: ± 1 d, Eingangswid.: 10^{12}, Asymmetrie-Korrektur, Temperatur-Kompensation 0–100 °C, Batt.-Betrieb 9 Volt. LCD-Anzeige: 13 mm. Handl. Taschengehäuse. Betriebsfertig mit Batterie, Koffer, Elektrode „made in Germany", gereinigtes Wasser, Pufferlösung PH −0,7 + PH 4,0 in Kunststoff-Flaschen mit Dosierschlauch.
Best.-Nr. 0601566 ..... **329,– DM**

Völkner electronic

Postfach 5320 · 3300 Braunschweig
Tel. (0531) 8762-111 · Telex 952547

3½-Zoll-Version von Lotus 1-2-3, Open Access und Symphony

Für den Toshiba-Compact-PC kündigte Lotus für Ende November eine deutsche Ausführung des Tabellenkalkulations-Programms Lotus 1-2-3 auf 3½-Zoll-Disketten in der Version 2.01 an. Dieses Programm, von dem über zwei Millionen Exemplare verkauft worden sein sollen, läuft auf den Toshiba-Computern T3100, T2100 und T1100. Auch das integrierte Paket Symphony soll in der Version 1.2 auf 3½-Zoll-Disketten erhältlich sein. Mit dieser neuesten Version von Symphony können Daten von Open Access und Multiplan in Lotus-Format konvertiert werden. Auch das integrierte Paket Open Access II wurde von SPI auf 3½-Zoll-Disketten angekündigt. Die netzwerkfähige Version Open Access II Net, die für das Novell-Netz eingerichtet ist, kostet etwa 1800 DM.

Bauteiledaten auf Diskette

Die amerikanische D.A.T.A. Inc., die in der Bundesrepublik Deutschland durch den IWT-Verlag, Vaterstetten, vertreten ist, gibt eine Reihe von Elektronik-Nachschlagewerken heraus. Jetzt beginnt das Unternehmen damit, Disketten für PCs mit Bauelementedaten zu vertreiben. Zunächst sind Disketten über RAMs, ROMs und PLAs erhältlich, die jeweils etwa 320 DM kosten.
Will man z. B. die Kennwerte eines bestimmten RAMs suchen, dann gibt man die Bauteilenummer (z. B. 6116) ein, und der Computer bringt eine Auswahl mit den genauen Typenbezeichnungen auf den Schirm. Jetzt wählt man mit dem Cursor ein Element aus und erhält sofort die zugehörigen Werte. Das System läßt sich auch mit Eckdaten füttern und sucht dann diejenigen Bauteile heraus, die den Anforderungen genügen. Bei RAMs kann man folgende Parameter vorgeben:

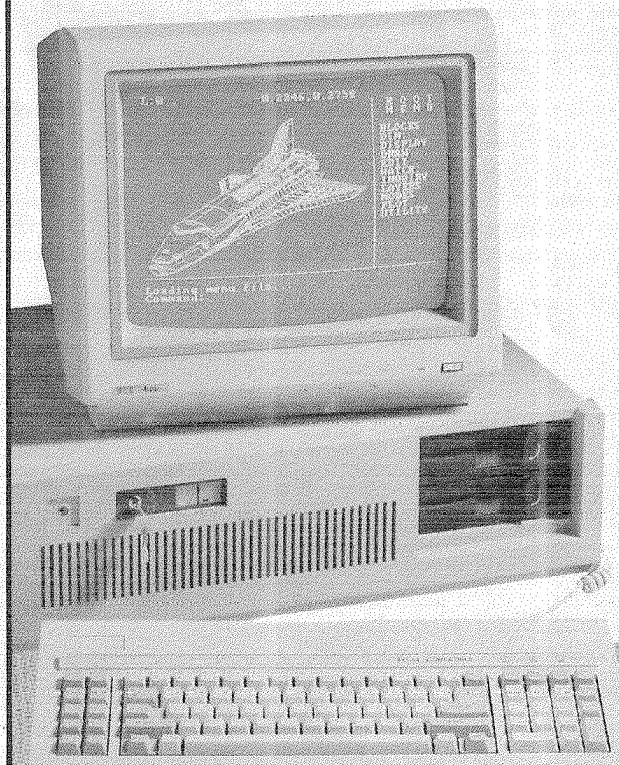
- Bauteiletyp (stat. RAM usw.),
- Organisation,
- Technologie,
- maximale Zugriffszeit,
- Betriebstemperaturbereich.

Querverweise auf die entsprechenden Datenbücher erleichtern das Auffinden von Anschlußbelegungen, Blockschaltbildern und Gehäusebezeichnungen. Zum Betrieb der Datenbanken braucht man einen IBM-PC mit mindestens 512 KByte RAM oder einen AT sowie ein Plattenlaufwerk mit mindestens 10 MByte Kapazität.

Vom FTZ abgesegnet

Mehrere Computertypen der Firma Sanyo dürfen zur Datenfernübertragung über Telefonleitungen der Deutschen Bundespost eingesetzt werden. Es handelt sich um die portablen Geräte mit Monochrom-Bildschirm der Serie MBC 670, die portablen Geräte mit 9-Zoll-Farbildschirm der Serie MBC 775 und die Tischgeräte der Serie MBC 890. Sie alle haben eine FTZ-Nummer erhalten.

10MHz PC/AT SYSTEM



Technische Daten des PC/AT:
10 MHz Hauptplatine, 640 KByte RAM
Monochrome Grafik-/Printerkarte
Laufwerk mit 1,2 MByte FDD
Festplatte mit 20 MByte, 65 m/s HDD
FDD/HDD Controllerkarte: Bis zu zwei FDD und zwei HDD
Netzteil: 200 W, 110/220 V
Einschubgehäuse, Platz für fünf Laufwerke
Englische Tastatur mit 84 Tasten

Erweiterungen:
RAM: 1 MByte RAM auf Hauptplatine
2,5 MByte RAM-Karte
Anderssprachige Tastatur mit 95, 100, 105 Tasten
Bis zu zwei Laufwerke (360 KByte oder 1,2 MByte)
Grafikkarte: MGP, CGP, EGA, HEGA
Festplatte: 20–85 MByte
Backup: 40 MByte Tape Streamer
I/O: Seriell-/Parallelkarte
3 MByte Multifunktionskarte
Netzwerk: Auf Anfrage

FOXEN CO., LTD.
Rm. 508, No. 432, Sec. 1, Keelung Rd., Taipei, Taiwan, R.O.C.
Telex: 29262 FOXEN Tel: (02) 709-9051, 776-2537

Mikroelektronik-Ausbildung, komplett: DM 498,–

Unser neues Einsteigerpaket
(Best.-Nr. 10 870) beinhaltet:

Funktionsbereites Computer-System
auf Z 80 Basis, fast beliebig erweiterbar.

Vier Kurse „Mikroelektronik-Einführung" von Christiani –
eine fundierte Einführung in die Mikroelektronik –
speziell für Anfänger geschrieben!

Den Einstieg in ein modulares Computersystem, den aus dem Fernsehen bekannten „NDR-Computer"; ausbaubar bis zum 8 bit CP/M 2.2-System mit Festplatte oder zum 32 bit 68020-System.

Interessiert? Anhängende Bestellkarte ausfüllen:

Best.-Nr.	Bezeichnung	Preis (incl. MWSt.)
10 870	**Komplettpaket Einsteiger**	**DM 498,–**
10 834	Farbkatalog über das Gesamtsystem 168 Seiten, Schutzgebühr	DM 10,–
10 061	Abonnement LOOP, die Zeitung für alle Z 80- und 680xx-Anwender, 5 Ausgaben	DM 20,–

GRAF computer

NDR-Computer
mc-CP/M-Computer

HANNOVER MESSE
CeBIT'87
4. – 11. MÄRZ 1987
Halle 6 F 21

GRAF ELEKTRONIK SYSTEME GMBH
Magnusstraße 13 · Postfach 1910 · 8960 Kempten (Allgäu) · Telefon: (0831) 5211
Telebox: 831804=GRAF · Telex: 17831804=GRAF · Datentelefon: (0831) 69130

Filiale Hamburg:
Ehrenbergstraße 16
2000 Hamburg 50
Telefon (0 40) 38 81 51

Filiale München:
Georgenstraße 61
8000 München 40
Telefon (0 89) 2 71 58 58

Händler:
Jörg Kanß · 1000 Berlin 30 · Telefon (0 30) 2 62 74 74
GMCP oHG · 2800 Bremen 33 · Telefon (04 21) 21 10 88
Schweiz: SYSTECH · CH-4104 Oberwil · Tel. (0 61) 35 10 31

Hardware-Erweiterungen für den Apple II

Zusatzschaltungen für Apple II, Apple IIe und kompatible Computer, Oskar Merker, 278 Seiten, Kartoniert, 48 DM, Franzis-Verlag, München
ISBN 3-7723-8271-1

Das hervorragende Buch von Oskar Merker ist eines der wenigen deutschsprachigen Bücher, die sich mit Hardware-Erweiterungen für die Apple II-Serie beschäftigt. Zuerst werden die Grundlagen, also der 6502-Prozessor, Computer allgemein und der Apple II im besonderen erklärt. Dabei werden die Pin-Belegungen des Handreglers und der Slots ausführlich erklärt. Es schließt sich die Beschreibung von Grundplatinen und auf diese aufbauende Platinen an. Der Verbindung von Hard- und Software ist ein eigenes Kapitel gewidmet. Im Gegensatz zu anderen Abhandlungen in Büchern oder Zeitschriften, in denen Schaltungen nur vorgestellt werden, erklärt der Autor die Schaltungen sehr genau und versäumt nicht, alle wichtigen ICs bis ins Detail vorzustellen. Der positive Eindruck, den dieses Buch hinterlassen hat, wird durch das breite Spektrum der beschriebenen Hardware verstärkt. Von der I/O-Platine über serielle und parallele Erweiterungen bis zu Uhr-, Ton- und Pseudo-ROM-Karten ist alles vorhanden. Ein Nachteil ist, daß für die Centronics- und RS232-Schnittstelle keine Standard-Pinbelegung angegeben wird. Das Buch ist sehr empfehlenswert und sein Geld wert. Die Schaltungen funktionieren mit Apple II+ und Apple IIe.

Wolfram Rittmeyer

Syntax-analyse

Otto Mayer, Syntaxanalyse, Reihe Informatik, Band 27, 434 Seiten, Kartoniert, 39 DM, Bibiliographisches Institut, 1986, Mannheim
ISBN 3-411-03139-5

Das Buch „Syntaxanalyse" entstand aus Vorlesungen, die Prof. Mayer an der Universität Kaiserslautern gehalten hat – die primäre Zielgruppe sind also Studenten der Informatik. Aber schließlich sind alle in der Computerbranche bis zur Pensionierung in gewisser Weise „notorische Studenten". Das Buch führt in die aktuellen Methoden der Syntaxanalyse ein. Es werden die allgemeinen Prinzipien der Analyseverfahren in Hinblick auf den praktischen Einsatz bei der Übersetzung von Programmiersprachen behandelt. Natürlich wollen die wenigsten von Ihnen einen Compiler für eine Programmiersprache schreiben und für das Programmieren eines einfachen Kommandointerpreters braucht man kein Buch aus dem Bereich des Compilerbaus. Für wen, außer den oben erwähnten Studenten, eignet sich das Buch dann noch? Eigentlich für alle, die einen Interpreter oder Compiler für eine Spezialsprache schreiben müssen. Ich denke da zum Beispiel an die Verbindung Künstliche Intelligenz und Robotersteuerung sowie alle verwandten Bereiche. Immer wenn es notwendig ist, eine Folge von Befehlsworten zu untersuchen und in bestimmte Aktionen umzusetzen, kommen die in diesem Buch dargestellten Prinzipien zum tragen. Das Buch bietet dafür die theoretischen Grundlagen, die Umsetzung in ein Programm bleibt dem Leser überlassen. JPL

Datenübermittlung

Walter Tietz, CCITT-Empfehlungen der V-Serie und der X-Serie, Band 3: Datenübermittlungsnetze – Schnittstellen, Kart., 554 Seiten, 168 DM, Decker's Verlag, G. Schenk GmbH, Heidelberg.
ISBN 3-7685-0286-4

Der Verlag ist dem Insider schon lange als Spezialist für Normen und Richtlinien der elektronischen Datenkommunikation bekannt. Das Buch von Walter Tietz ist der dritte Band einer Reihe, der jetzt in der fünften, erweiterten Auflage erschienen ist. Band 1 der Reihe „CCITT-Empfehlungen der V-Serie und der X-Serie" beschäftigt sich mit der Datenübertragung im Telefonnetz (z. B. Modems), der zweite Band behandelt die Datenübermittlungsnetze von der Seite der verfügbaren Dienste und ihrer Leistungsmerkmale her. Der dritte Band beschäftigt sich mit den Schnittstellendefinitionen der Empfehlungen X.20 bis X.32 für die Datenübertragung in öffentlichen Netzen. In diesen Definitionen sind alle Parameter für die Datenübertragung beschrieben und festgelegt: elektrische Eigenschaften, Störsicherheit und natürlich die Datenformate und Quittungsprotokolle bei der gesicherten Übertragung von Datenpaketen. Die CCITT-Empfehlungen sind sicher keine Bücher für den Freak, der mal schnell per Akustikkoppler eine Mailbox anruft. Für den DFÜ-Profi ist die Reihe ein wichtiges und unabdingbares Arbeitsmittel. JPL

Prolog

Einführung in die Programmierpraxis. Von Peter Schnupp. 315 Seiten. Geb. 48 DM. Carl Hanser Verlag, München, Wien.
ISBN 3-446-14582-6

Ein Buch, mit dem der Praktiker Prolog lernen kann: Grundlagen werden kurz und mit Beispielen vermittelt, die sofort zu eigenen Experimenten herausfordern. Die Schwierigkeit, unterschiedlichen Implementierungen von Prolog gerecht zu werden, ist hier gut gelöst: Auf über 50 Seiten werden alle verwendeten Prädikate mit Namen, Syntax, Beschreibung, Beispiel und Querverweisen zu ähnlichen Prädikaten behandelt. Damit wird es zum lehrreichen Vergnügen, die vorgestellten Beispiele auf einen anderen Interpreter bzw. Compiler zu übertragen, solange diese dem Clocksin-Mellish-Standard folgen – anders ausgedrückt: Turbo-Prolog-Anwender sollten sich lieber an ihr Handbuch halten, da dieses ähnliche Beispiele bietet. Einen Rechner mit Prolog sollte man allerdings bei der Lektüre immer in Reichweite haben.
Angenehm ist, daß sich die Beispiele nicht in Verwandtschafts- und Abstammungsbeziehungen erschöpfen (obwohl auch diese vorkommen): Neben einer Datenbank-Simulation findet man ein modulares, kleines und damit überschaubares Expertensystem, das auch eine in der Wissensbasis festgelegte Erklärung der gestellten Fragen vorsieht. Und der Umstand, daß man nach Lektüre des Buches an diesem System so manche Verbesserung, Änderung oder Erweiterung vornehmen wird, spricht eigentlich auch nur für seinen didaktischen Wert. Kr.

Dieter Strauß

Der gläserne PC-Bus

Dem PC-Bus auf die Signale geschaut

Auf dem Markt werden viele PC-Erweiterungskarten angeboten. Wie der PC-Bus funktioniert und wie man für diesen Bus Karten selbst „stricken" kann, darüber gibt es wenig Informationen. mc klärt die Bedeutung der einzelnen Signale und zeigt an einem Beispiel wie man sich an den PC-Bus ankoppeln kann.

Im Standard-PC mit dem Prozessor 8088 gibt es zwei Bussysteme: den lokalen Prozessorbus und den Systembus, auch kurz PC-Bus genannt. Am lokalen Bus sind neben dem Prozessor selbst der Bus-Controller 8288, der optionale Numerik-Prozessor 8087, der Interrupt-Controller 8259A, drei 8-Bit-D-Latches 74LS373, ein Bustreiber 74LS244 und der bidirektionale Bustreiber 74LS245 angeschlossen. Die drei 8-Bit-D-Latches gewinnen aus der zeitgemultiplexten Adreß-/Dateninformation des lokalen Prozessorbusses eine auf dem PC-Bus stabil anstehende Adresse. Da beim Standard-PC der Prozessor im Maximal-Modus betrieben wird, erzeugt nicht der Prozessor, sondern der Bus-Controller (8288) die Bussteuersignale. Der Bus-Controller gewinnt aus den drei vom 8088 bereitgestellten Statussignalen (S0...S2), dem Systemtakt (Taktgenerator 8284) und den von einer Zusatzlogik erzeugten Signalen CEN (Command Enable) und AEN (Address Enable) die Steuersignale des PC-Busses. Der lokale Prozessorbus ist nur auf der Systemleiterplatte vorhanden und deshalb vom Benutzer nicht zugreifbar.

Tabelle 1: Anschlußbelegung der Erweiterungssteckplätze

Signalname	Stiftbezeichnung		Signalname
GND	B01	A01	$\overline{\text{I/O CHCK}}$
RESET	B02	A02	D7
+5 V	B03	A03	D6
IRQ2	B04	A04	D5
–5 V	B05	A05	D4
DREQ2	B06	A06	D3
–12 V	B07	A07	D2
reserviert	B08	A08	D1
+12 V	B09	A09	D0
GND	B10	A10	I/O CHRDY
$\overline{\text{MEMW}}$	B11	A11	AEN
$\overline{\text{MEMR}}$	B12	A12	A19
$\overline{\text{IOWC}}$	B13	A13	A18
$\overline{\text{IORC}}$	B14	A14	A17
$\overline{\text{DACK3}}$	B15	A15	A16
DREQ3	B16	A16	A15
$\overline{\text{DACK1}}$	B17	A17	A14
DREQ1	B18	A18	A13
$\overline{\text{DACK0}}$	B19	A19	A12
CLK	B20	A20	A11
IRQ7	B21	A21	A10
IRQ6	B22	A22	A9
IRQ5	B23	A23	A8
IRQ4	B24	A24	A7
IRQ3	B25	A25	A6
$\overline{\text{DACK2}}$	B26	A26	A5
TC	B27	A27	A4
ALE	B28	A28	A3
+5 V	B29	A29	A2
OSC	B30	A30	A1
GND	B31	A31	A0

Die auf den ersten Blick umständlich aussehende Lösung mit zwei Bussen, einen separaten Bus-Controller und Latches zum Demultiplexen wurde von den Entwicklern der 8088/8086-Familie gewählt, um den Systementwicklern möglichst viel Freiheiten beim Systementwurf zu geben und um diese Bausteine in Standard-DIL-Gehäusen unterbringen zu können. Von den Möglichkeiten der Oberflächenmontagetechnik, die hochpolige Bausteine in kleine Gehäuse unterbringt, war man zum Zeitpunkt des Entwurfs der 8088/8086-Familie noch weit entfernt.

Im PC kann man zwei Buszyklustypen unterscheiden – vom Prozessor 8088 und vom DMA-Controller 8237-5 initiierte Buszyklen. Der Prozessor kann fünf verschiedene Buszyklen ausführen: Lese- und Schreibzyklen im Speicher- und im E/A-Bereich sowie Interrupt-Quittungszyklen. Der DMA-Controller kann nur drei verschiedene Buszyklen ausführen.

Er kann Daten vom Speicher zum E/A-Port oder umgekehrt „schaufeln" und den Speicher-Refresh mit einem Pseudo-Lesezugriff durchführen. Im Standard-PC sind sowohl die Speicher-Lese- als auch die Speicher-Schreib-Zyklen vier Prozessortakte lang. Da bei 4,77 MHz ein Prozessortakt 210 ns dauert, sind diese Buszyklen 840 ns lang. Die E/A-Port-Buszyklen könnten genauso schnell sein, die Entwickler des PCs haben jedoch einen Wartetakt eingefügt, so daß für diese Buszyklen fünf Prozessortakte (1050 ns) gebraucht werden.

Im PC dauert der Pseudo-Lesezugriff zum Speicher-Refresh fünf Prozessortakte, die DMA-Speicher-Schreib- und

DMA-Speicher-Lese-Zyklen sechs Prozessortakte. Während DMA-Zyklen befinden sich nie E/A-Port-Adressen auf dem Adreßbus! Die E/A-Ports werden statt dessen durch die DACK0...DACK3-Leitungen angesprochen.

Die 62 Pole des Erweiterungssteckers

Der PC-Bus besteht aus 20 Adreß-, 8 Daten-, 19 Steuer-, 6 Interrupt- und 8 Leitungen für die Stromversorgung. Alle Signale entsprechen dem TTL-Pegel und sind nicht gemultiplext. Die Signale des PC-Busses sind an den Bus-Erweiterungssteckplätzen verfügbar.

Auf der Systemleiterplatte werden Direktsteckverbinder zur Aufnahme der Erweiterungskarten eingesetzt. Diese Steckverbinder sind 62polig. Ein Pol ist nicht belegt. Eine Seite wird A, die andere B genannt. Die Anschlüsse werden mit A01...A31 und B01...B32 bezeichnet (*Tabelle* 1). A1 und B1 liegen auf der Seite des Steckverbinders, die zur Gehäuse-Rückwand zeigt.

Um die Wirkungsweise des PC-Busses zu verstehen, muß man die Bedeutung der 19 Steuersignale kennen. Danach bereitet es keine größeren Schwierigkeiten mehr selbst eine Busanpassung vorzunehmen und eine Karte, die nicht auf dem Markt erhältlich ist, zu entwerfen.

Die Frequenz des Oszillatorsignals OSC beträgt bei Systemen mit 4,77 MHz Systemtakt 14,31818 MHz. Die Periodendauer ist 70 ns, das Tastverhältnis 1:1. Der Systemtakt CLK ist der durch drei geteilte Oszillatortakt. Das Tastverhältnis ist 1:2. Das RESET-Signal (RESET DRV) setzt alle Erweiterungskarten in einen definierten Anfangszustand. Dieses Signal wird als Power-On-Reset benutzt. Weicht während des Betriebs eine Versorgungsspannung von dem vorgeschriebenen Wert ab, so wird dieses Signal ebenso aktiviert. Das Speicher-Lese-Signal ($\overline{\text{MEMR}}$ = Memory Read) und das Speicher-Schreib-Signal ($\overline{\text{MEMWR}}$ = Memory Write) wird entweder vom Bus- oder vom DMA-Controller erzeugt. Daten müssen bis spätestens 30 ns vor der ansteigenden Flanke von $\overline{\text{MEMR}}$ auf den Bus (D0...D7) gelegt werden. Daten sind spätestens kurz vor der ansteigenden Flanke von $\overline{\text{MEMWR}}$ gültig.

Lese- bzw. Schreibbefehle für den E/A-Bereich sind $\overline{\text{IORC}}$ (I/O Read Command) und $\overline{\text{IOWC}}$ (I/O Write Command). Auch diese Signale können vom Bus-Controller oder vom DMA-Controller erzeugt werden. Bis spätestens 30 ns vor der ansteigenden Flanke von $\overline{\text{IORC}}$, muß der adressierte E/A-Port sein Datum auf den Bus gelegt haben. Initiert der DMA-Controller einen Lesezyklus, so liegt auf dem Adreßbus keine E/A-Adresse, sondern die Speicheradresse, in die der Inhalt des E/A-Ports geschrieben werden soll.

E/A-Ports sollten die auf D0...D7 anstehenden Daten mit der ansteigenden Flanke von $\overline{\text{IOWC}}$ übernehmen.
ALE (Address Latch Enable) wird vom 8288 (Bus-Controller) erzeugt. Mit ALE wird jeder vom Prozessor eingeleitete Buszyklus signalisiert. In Verbindung mit den Adreßleitungen (A0...A19) ist das ALE-Signal von besonderer Bedeutung. Auf der Systemleiterplatte wird dieses Signal zum Demultiplexen der lokalen Adreß-/Datenbussignale des 8088 verwendet. Kurz vor der fallenden Flanke von ALE ist die auf dem Adreßbus liegende Adresse gültig. Während DMA-Zyklen wird dieses Signal nicht generiert.

Direkter Speicherzugriff

Die Signale AEN (Address Enable), TC (Terminal Count), DRQ1...DRQ3 (DMA Request) und $\overline{\text{DACK0}}$...$\overline{\text{DACK3}}$ (DMA Acknowledge) betreffen den DMA-Controller. Mit dem AEN-Signal (Address Enable) wird ein laufender DMA-Zyklus angezeigt. Der Prozessor hat den Bus an den DMA-Controller übergeben. Da im DMA-Betrieb $\overline{\text{IOWC}}$ und $\overline{\text{IORC}}$ aktiviert sein können, obwohl eine Speicheradresse auf dem Bus ist, sollte AEN zur Decodierung von E/A-Portadressen mit verwendet werden. Will ein E/A-Kanal Daten mit dem Speicher austauschen und die CPU umgehen, so muß dieser ein DMA-Anforderungssignal aktivieren. DMA-Anforderungen mit DRQ1...DRQ3 (DMA Request) bleiben so lange aktiv bis die Peripherie das Quittungssignal $\overline{\text{DACK0}}$...$\overline{\text{DACK3}}$ vom DMA-Controller erhält. DRQ1 hat die höchste, DRQ3 die niedrigste Priorität. Ist $\overline{\text{DACK0}}$ aktiv, so läuft gerade ein Refresh der dynamischen RAMs ab. Auf dem Adreßbus stehen Refresh-Adressen an. Der $\overline{\text{DACK0}}$-Refresh wiederholt sich alle 72 Takte. DMA-Kanal 2 wird vom Floppy-Controller und DMA-Kanal 3 vom Festplatten-Controller benutzt.

DMA-Kanal 1 ist für spätere Erweiterungen reserviert.

Der DMA-Controller gibt auf die TC-Leitung einen Impuls, sobald einer der vier DMA-Kanäle seine programmierte Anzahl an DMA-Zyklen erreicht hat. Um festzustellen, welcher der DMA-Kanäle fertig ist, müssen TC und die $\overline{\text{DACK0}}$-Signale mit Und-Gliedern verknüpft werden.

Mit $\overline{\text{I/O CHCK}}$ (I/O Channel Check) signalisieren externe Speicher einen aufgetretenen Paritätsfehler an. Wenn dieses Signal gesetzt ist, wird ein nichtmaskierbarer Interrupt (NMI) erzeugt.

Mit I/O CHRDY (I/O Channel Ready) lassen sich Buszyklen verlängern. Langsame E/A-Ports oder Speicher, die nicht während der normalen Buszyklen von vier Takten auf Lese- oder Schreibbefehle reagieren können, müssen dieses Signal verwenden. Mit der negativen Flan-

Tabelle 2: Portadressen

E/A-Adresse	A9	Funktion
0000H...000FH	0	DMA-Controller (8237A-5)
0020H...0021H	0	Interrupt-Controller (8259A-5)
0040H...0043H	0	Zeitgeber/Zähler (8253-5)
0060H...0063H	0	Systemregister (8255A-5)
0080H...0083H	0	DMA-Seitenregister (74LS670)
00A0H	0	NMI-Maskierungs-Bit
0201H	1	Spiele (Game Port)
0278H...027FH	1	zweiter Drucker
0278H...02FFH	1	zweite serielle Schnittstelle
0300H...031FH	1	Prototypkarte
0320H...032FH	1	Festplatten-Controller
0378H...037FH	1	Druckerschnittstelle (parallel)
03B0H...03BFH	1	Monochromadapter und Drucker
03D0H...03DFH	1	Farbgrafikkarte
03F0H...03F7H	1	Floppy-Controller
03F8H...03FFH	1	serielle Schnittstelle

ke (Wechsel von High auf Low) von I/O CHRDY läßt sich die CPU innerhalb von Buszyklen anhalten. Alle auf dem Adreß-, Daten- und Steuerbus anstehenden Daten bleiben unverändert. Die CPU führt Wartezyklen aus. Maximal dürfen zehn Wartezyklen ausgeführt werden. I/O CHRDY muß mit offenen Kollektor-Treibern angesteuert werden (verdrahtetes Oder).

Fünf IRQ-Leitungen

Für Unterbrechungsanforderungen werden die Leitungen IRQ2...IRQ7 (Interrupt Request) benutzt. IRQ2 hat die höchste und IRQ7 die niedrigste Priorität. Die höherprioren Interrupts IRQ0 und IRQ1 sind nicht auf die Erweiterungssteckplätze geführt. Mit der ansteigenden Flanke an einem der IRQ-Leitungen wird ein Interrupt erzeugt. Ein Interrupt wird vom Prozessor in der Interrupt-Service-Routine quittiert. Wenn die Unterbrechungsanforderung nicht so lange aufrecht erhalten bleibt, bis diese vom Prozessor quittiert wird, so wird ein Interrupt mit der niedrigsten Priorität angenommen.

Am Interrupt-Contoller (8259A) sind die acht Leitungen IRQ0...IRQ7 angeschlossen. IRQ0 ist mit dem Kanal 0 des Zähler-/Zeitgeberbausteins 8253-5 verbunden. Der Ausgang des Kanal 0 erzeugt ein Rechtecksignal von 18,2 Hz und löst damit alle 54,94 ms einen Interrupt IRQ0 aus. Dieser wird vom BIOS für die interne Uhr verwendet. IRQ1 wird von der Tastatur, IRQ 4 von der seriellen Schnittstelle, IRQ6 vom Floppy-Controller und IRQ7 vom parallelen Drucker-Port verwendet. IRQ2, IRQ3 und IRQ5 sind frei für zukünftige Erweiterungen.

IRQ3 kann z. B. für eine zweite serielle Schnittstelle verwendet werden. Der nicht auf die Erweiterungssteckplätze herausgeführte NMI (nicht maskierbarer Interrupt) wird bei RAM-Paritätsfehlern auf der Systemleiterplatte, vom Numerik-Prozessor 8087 und vom $\overline{\text{I/O-CHCK}}$-Signal gesetzt.

Auf dem PC-Bus stehen vier Spannungen zur Verfügung: +5 V, –5 V, +12 V und –12 V. Auf Massepotential liegen drei Leitungen.

Der E/A-Adreßraum

Der Prozessor 8088 unterstützt 65536 E/A-Portadressen. Da im E/A-Bereich des PCs jedoch nur die Adreßleitungen A0...A9 benutzt werden, sind nur 1024 E/A-Portadressen ansprechbar.

Die ersten 512 E/A-Portadressen (000H...1FFH) sind auf der Systemleiterplatte (A9 = 0), die restlichen 512 auf Erweiterungskarten verfügbar (*Tabelle 2*).

Bild 1 zeigt eine einfache Decodierschaltung für den PC-Bus. Der Komparator 74LS688 vergleicht die auf A3...A8 anstehenden Adressen mit der am DIP-Schalter eingestellten Adresse. Steht beim Entwurf der Erweiterungskarte die Portadresse bereits fest, so kann der DIP-Schalter weggelassen und die Anschlüsse Q0...Q5 des 74LS688 fest verdrahtet werden. Damit während DMA-Zyklen keine Adressengleichheit angezeigt wird, wird AEN zur Qualifizierung des $\overline{\text{P=Q}}$-Signals mit verwendet. Durch den Vergleich von A9 mit einer logischen Eins wird sichergestellt, daß ein bereits auf der Systemleiterplatte vorhandenes E/A-Port auf der Erweiterungskarte nicht nochmals decodiert wird. Der Bus-Transceiver 74LS245 wird erst selektiert, wenn $\overline{\text{IOWC}}$ oder $\overline{\text{IORC}}$ und $\overline{\text{P=Q}}$ aktiv sind. $\overline{\text{IORC}}$ schaltet die Richtung des Datentransfers um. Das $\overline{\text{CS}}$-Signal ist in einem E/A-Adreßbereich von 8 Adressen aktiv, da A0...A2 nicht decodiert sind. Es ist zu beachten, daß $\overline{\text{CS}}$ auch von Speicher-Adressen aktiviert wird. Die weitere Verschaltung von $\overline{\text{CS}}$ hängt jedoch stark von der nachfolgenden Schaltung ab und kann daher nicht allgemein beschrieben werden. Mit der einfachen Decodierschaltung und den Kenntnissen der PC-Bus-Signale dürfte es jetzt einfacher sein, selbst Erweiterungskarten zu entwerfen. Im PC-Bereich lohnt sich der Entwurf von Erweiterungskarten nur, wenn diese Karten nicht bereits „spottbillig" auf dem Markt angeboten werden. mc wird in Zukunft Bauanleitungen für spezielle PC-Erweiterungskarten veröffentlichen.

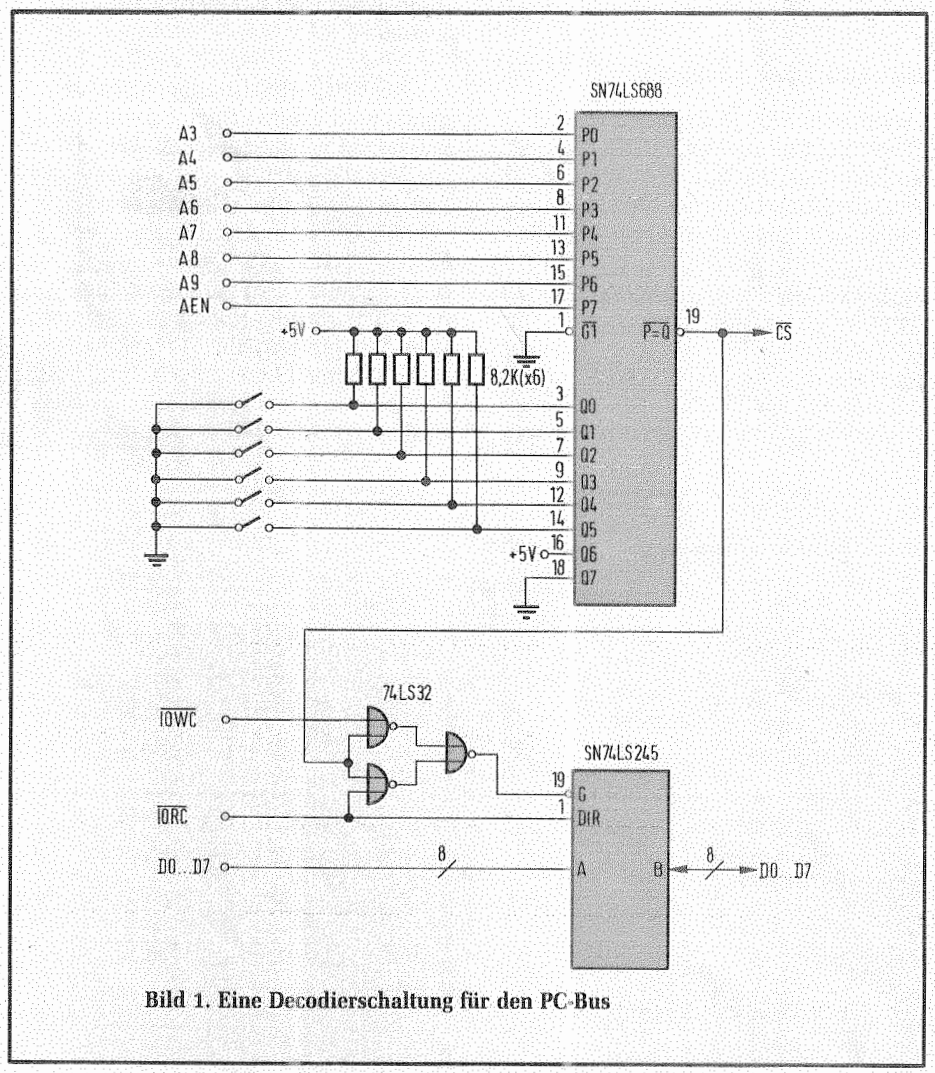


Bild 1. Eine Decodierschaltung für den PC-Bus

Frank Brendle

EGA: Print-Screen-Funktion

Bildschirm-Speicher der EGA-Karte verzerrungsarm ausdrucken

Die EGA-Karte (Enhanced Graphics Adapter) erweitert die Grafik-Fähigkeiten des IBM-PC erheblich. Leider gibt es für sie noch wenig Software. mc zeigt wie die Print-Screen-Funktion mit der EGA-Karte durchgeführt werden kann.

Das Programm GREGA (*Bild 1*) ergänzt das MS-DOS-Programm GRAPHICS. Bevor die Print-Screen-Funktion mit GREGA aufgerufen werden kann, muß zuerst GRAPHICS gestartet werden. Nach Drücken der Tastenkombination „Shift-PrtSc" erwartet der Computer eine weitere Eingabe. Wird „F1" gedrückt, so gibt der Drucker den Inhalt der ersten Farb-Ebene normal aus. „F2" bewirkt, daß das Bild invers gedruckt wird. Jede andere Taste ruft die ursprüngliche Druck-Routine auf.

Im Programm wird der Interrupt-Vektor 5H (BIOS-Bildschirm-Ausdruckroutine) auf den Anfang des neuen Druck-Programms umgebogen. Mit dem Interrupt 27H wird das im Speicher vorhandene Programm beendet und ins DOS zurückgesprungen. Das Programm bleibt jedoch im Speicher. Das Schirmbild wird aus Formatgründen um 90 Grad gedreht ausgedruckt (*Bild 2*). Interrupt 17H ruft die BIOS-Ausdruckroutinen auf. Das ausgedruckte Bild ist 148 mm × 226 mm groß. Das Seitenverhältnis weicht nur geringfügig von dem des Schirmbildes ab.

GREGA wurde für einen Drucker Epson FX-85 geschrieben. Da nur Standard-Escape-Sequenzen benutzt werden, dürfte der Betrieb anderer Geräte keine Schwierigkeiten machen.

```
57B6:0100 31C0          XOR     AX,AX       ; Anfangsadresse des alten Druck-
57B6:0102 8ED8          MOV     DS,AX       ; Programms retten.
57B6:0104 A11400        MOV     AX,[0014]   ;
57B6:0107 2E            CS:                 ;
57B6:0108 A33B01        MOV     [013B],AX   ;
57B6:010B A11600        MOV     AX,[0016]   ;
57B6:010E 2E            CS:                 ;
57B6:010F A33D01        MOV     [013D],AX   ;
57B6:0112 B82101        MOV     AX,0121     ; Der Unterbrechungsvektor 5 zeigt
57B6:0115 A31400        MOV     [0014],AX   ; nun auf den Anfang des neuen
57B6:0118 8C0E1600      MOV     [0016],CS   ; Druck-Programms.
57B6:011C BAB701        MOV     DX,01B7     ; Neue Routine resident machen.
57B6:011F CD27          INT     27          ; Zurück ins DOS.
57B6:0121 FB            STI                 ; Interrupts wieder zulassen.
57B6:0122 50            PUSH    AX          ; AX retten.
57B6:0123 B400          MOV     AH,00       ; Ein Zeichen von der Tastatur
57B6:0125 CD16          INT     16          ; einlesen.
57B6:0127 80FC3B        CMP     AH,3B       ; Wurde F1 gedrückt?
57B6:012A 7504          JNZ     0130        ; Wenn nein, weiterprüfen.
57B6:012C B0FF          MOV     AL,FF       ; Wert für normalen Ausdruck.
57B6:012E EB0F          JMP     013F        ; Bildschirminhalt drucken.
57B6:0130 80FC3C        CMP     AH,3C       ; Wurde F2 gedrückt?
57B6:0133 7504          JNZ     0139        ; Wenn nein, altes Druck-Programm
                                            ; aufrufen.
57B6:0135 B000          MOV     AL,00       ; Wert für invertierten Ausdruck.
57B6:0137 EB06          JMP     013F        ; Bildschirminhalt drucken.
57B6:0139 58            POP     AX          ; AX wiederherstellen.
57B6:013A EA00000000    JMP     0000:0000   ; Altes Druck-Programm aufrufen.
57B6:013F 2E            CS:                 ; Modus (inverse/normal) setzen.
57B6:0140 A26D01        MOV     [016D],AL   ;
57B6:0143 53            PUSH    BX          ; BX retten.
57B6:0144 51            PUSH    CX          ; CX retten.
57B6:0145 52            PUSH    DX          ; DX retten.
57B6:0146 56            PUSH    SI          ; SI retten.
57B6:0147 1E            PUSH    DS          ; DS retten.
57B6:0148 FC            CLD                 ; Richtungsflagge löschen.
57B6:0149 B800A0        MOV     AX,A000     ; Segment des Bildschirmspeicher-
57B6:014C 8ED8          MOV     DS,AX       ; Anfangs nach DS schreiben.
57B6:014E 31D2          XOR     DX,DX       ; DX löschen -) Drucker 0 wird
                                            ; verwendet.
57B6:0150 BE9601        MOV     SI,0196     ; Drucker auf 8/72 Zoll Zeilenab-
57B6:0153 E83300        CALL    0189        ; stand schalten.
57B6:0156 BB5000        MOV     BX,0050     ; Zeilenzähler aufsetzen.
57B6:0159 BEA501        MOV     SI,01A5     ; Drucker auf Grafik schalten und
57B6:015C E82A00        CALL    0189        ; Druckkopf nach rechts rücken.
57B6:015F B95E01        MOV     CX,015E     ; Spaltenzähler aufsetzen.
57B6:0162 B85000        MOV     AX,0050     ; Spalte mit 80 multiplizieren.
57B6:0165 F7E1          MUL     CX          ;
57B6:0167 29D8          SUB     AX,BX       ; Zeilennummer subtrahieren.
57B6:0169 89C6          MOV     SI,AX       ; Adresse des zu druckenden Bytes
                                            ; nach SI schreiben.
57B6:016B AC            LODSB               ; Zu druckendes Byte nach AL.
57B6:016C 3400          XOR     AL,00       ; Evtl. AL invertieren.
57B6:016E B400          MOV     AH,00       ; Bitmuster in AL auf dem Drucker
57B6:0170 CD17          INT     17          ; ausgeben.
```

Bild 1. Mit diesem Programm läßt sich der Bildschirminhalt verzerrungsarm ausdrucken

```
57B6:0172 B400          MOV     AH,00       ; Bitmuster nochmals ausgeben (um
57B6:0174 CD17          INT     17          ; den Kontrast zu erhöhen).
57B6:0176 E2E9          LOOP    0162        ; Nächste Spalte drucken.
57B6:0178 FECB          DEC     BL          ; Zeilenzähler dekrementieren.
57B6:017A 75DC          JNZ     0159        ; Nächste Zeile drucken.
57B6:017C BE9C01        MOV     SI,019C     ; Drucker auf normalen Zeilenab-
57B6:017F E80700        CALL    0189        ; stand schalten.
57B6:0182 1F            POP     DS          ; DS wiederherstellen.
57B6:0183 5E            POP     SI          ; SI wiederherstellen.
57B6:0184 5A            POP     DX          ; DX wiederherstellen.
57B6:0185 59            POP     CX          ; CX wiederherstellen.
57B6:0186 5B            POP     BX          ; BX wiederherstellen.
57B6:0187 58            POP     AX          ; AX wiederherstellen.
57B6:0188 CF            IRET                ; Fertig.


;       Unterprogramm zur Ausgabe einer Zeichenkette an den Drucker.
;
;         Eingabe: SI = Anfang der Zeichenkette
;         Die Zeichenkette muß mit 0 abgeschlossen werden

57B6:0189 2E            CS:                 ; Zeichen holen.
57B6:018A AC            LODSB               ;
57B6:018B 08C0          OR      AL,AL       ; Ist es 0?
57B6:018D 7406          JZ      0195        ; Wenn ja, aufhören.
57B6:018F B400          MOV     AH,00       ; Zeichen auf dem Drucker ausge-
57B6:0191 CD17          INT     17          ; ben.
57B6:0193 EBF4          JMP     0189        ; Nächstes Zeichen ausgeben.
57B6:0195 C3            RET                 ; Ende.


;       Steuerzeichen-Tabelle für den Drucker

57B6:0196 1B4108        DB      1B,"A",08   ; Zeilenabstand = 8/72 Zoll.
57B6:0199 1B3200        DB      1B,"2",00   ; Neuen Zeilenabstand aktivieren.
57B6:019C 1B410C        DB      1B,"A",0C   ; Zeilenabstand =12/72 Zoll.
57B6:019F 1B32          DB      1B,"2"      ; Neuen Zeilenabstand aktivieren.
57B6:01A1 0D0A0A00      DB      0D,0A,0A,00 ; Wagenrücklauf, 2*Zeilenvorschub.
57B6:01A5 0D0A          DB      0D,0A       ; Wagenrücklauf, Zeilenvorschub.
57B6:01A7 202020        DB      20,20,20    ; Leerzeichen zum Einrücken des
57B6:01AA 202020        DB      20,20,20    ; Ausdrucks.
57B6:01AD 20202020      DB      20,20,20,20 ;
57B6:01B1 1B4CBC        DB      1B,"L",BC   ; Initialisierungszeichen für den
57B6:01B4 0200          DB      02,00       ; Grafik-Modus.
```

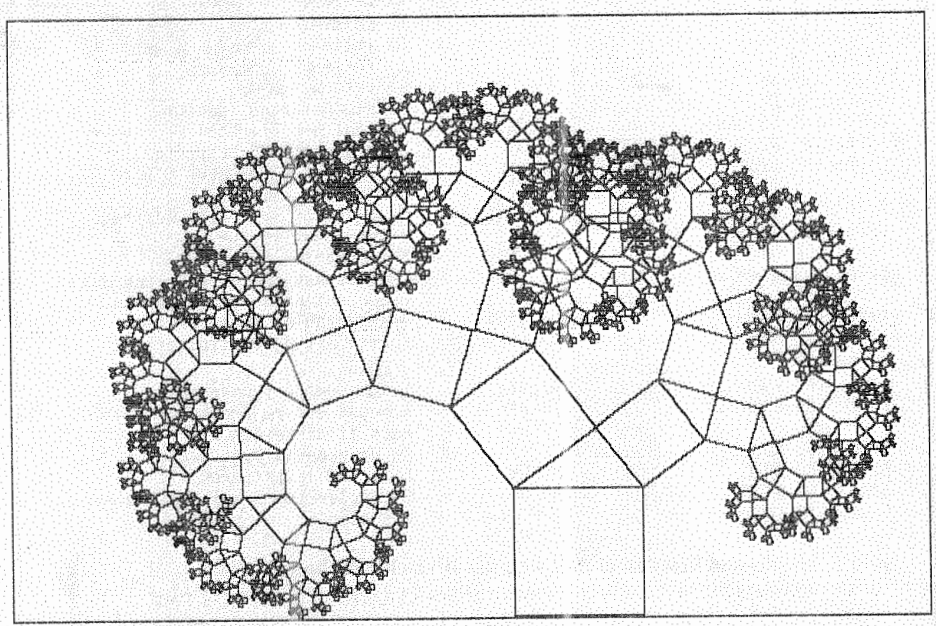


Bild 2. Der Baum des Pythagoras verzerrungsarm ausgedruckt

Telex mit PCs

Einen Durchbruch bei der Zulassung durch die Deutsche Bundespost schaffte jetzt die Firma Com-M-Tex aus der Schweiz: Auf bislang 40 MS-DOS-Rechnern darf nun mit postalischem Segen ein Programm laufen, welches den Computer zum Endgerät eines Telex-Anschlusses macht. Derart als Fernschreiber eingesetzt hat der PC Zugang zum weltweit größten Netz für schriftliche Übertragung.

Entscheidend für die Zulassung war das Urteil einer Experten-Kommision, wonach ein MS-DOS-Rechner eine Datenverarbeitungs-Anlage darstellt – wer hätte das gedacht?

So ganz trauen mag man allerdings bei der Post den kleinen Rechnern noch immer nicht. So müssen bislang mindestens zwei solche Computer über ein Netzwerk miteinander verbunden sein, der zweite Rechner kann dann bei Ausfall des ersten dessen Arbeit übernehmen. Wie die Post das aber nach der Abnahme noch prüfen will, bleibt ihr Geheimnis. Der am Telexhauptanschluß betriebene Rechner muß über eine Festplatte verfügen, und nach einem Netzausfall muß das Programm automatisch wieder gestartet werden. Der Anschluß des PC an das Telex-Netz erfolgt laut Zulassungs-Text über die EDF-Schnittstelle mit „zulässigen privaten Anschaltgeräten" und arbeitet mit dem Zeichensatz des CCITT-Codes Nr. 2.

Der Zugang zum Fernschreibnetz ist weitaus einfacher zu realisieren als der zu höheren Diensten, also etwa Datex-P. Denn auch einem normalen Fernschreiber kann ja zum Beispiel zwischendurch einfach mal das Papier ausgehen. Ein entscheidender Vorteil eines Rechners gegenüber einem herkömmlichen Fernschreiber liegt in der Möglichkeit, Texte direkt aus einem Textverarbeitungs-Programm zu überspielen, ein weiterer in der geringen Störanfälligkeit. Denn in den bisher üblichen Fernschreibern sind immer noch eine ganze Reihe von mechanischen Elementen zu finden, die erfahrungsgemäß für Verschleiß und Verschmutzung weit anfälliger sind als die Elektronik eines PC. Eben darum hat sich der Software-Hersteller nun zum Ziel gesetzt, auch die Zulassung einzelner MS-DOS-Rechner zu erreichen. In der Schweiz ist das schon heute möglich, aber dort war auch die jetzt in Deutschland genehmigte Lösung kein Problem.

Kr.

Stefan Peichl

Vier Bildschirmseiten

Speicherresidentes Programm unter MS-/PC-DOS

Der größte Teil der MS-/PC-DOS Anwender benutzt die IBM-Standard-Farbgrafikkarte entweder mit Schwarzweiß- oder Farbbildschirm im 80-spaltigen Textmodus. In diesen beiden Fällen ist der Bildschirmspeicher aber nur zu einem Viertel ausgenutzt. Das hier beschriebene kurze Programm ermöglicht die Nutzung des gesamten Bildschirmspeichers, so daß zwischen vier Bildschirmseiten umgeschaltet werden kann.

Das hier vorgestellte Programm *(Bild 1)* verbleibt nach dem Laden im Hauptspeicher und kann jederzeit über die Tastatur aktiviert werden. Es gibt einen guten Einblick in die Technik der Programmierung speicherresidenter Tools, die über Tastendruck gestartet werden.
Der Bildschirmspeicher auf der Farbgraphikkarte ist 16 KByte groß. Ein voller Bildschirm im 80-spaltigen Textmodus belegt aber nur 80 × 25 × 2 = 4000 Bytes, das sind genau 96 Bytes weniger als 4 KByte. Es passen zwar nur 2000 Zeichen auf einen Bildschirm, intern wird aber zu jedem Zeichen noch das sogenannte Attribut-Byte verwaltet, mit dem die Art der Darstellung gewählt werden kann. Daher der doppelte Speicherbedarf. Es bleiben also 12 KByte ungenutzt, die für drei weitere volle Bildschirmseiten reichten. Wozu sind diese restlichen 12 KByte da, wenn auch 4 KByte bereits genügen? Sie werden für den Grafikmodus gebraucht, in dem tatsächlich die Information einer Bildschirmseite 16 KByte belegt. Die Darstellungsform im Grafikmodus unterscheidet sich völlig von der des Textmodus. Es werden in beiden Modi sogar verschiedene Zeichen verwendet. Bekannt ist die Möglichkeit, bei gedrückter Alt-Taste eine Dezimalzahl auf dem abgesetzten Ziffernblock einzugeben. Mit dem Loslassen der Alt-Taste erscheint das der Dezimalzahl entsprechende ASCII-Zeichen auf dem Bildschirm. Nach dem Eintippen von Alt-130 müßte e-accent aigu zum Vorschein kommen. Lädt man nun ein echtes Grafikprogramm und gibt wieder Alt-130 ein, so erscheint ein kleines pointillistisches Bild, aber kein definiertes Zeichen. Im Grafikmodus gibt es nämlich nur die ersten 127 ASCII-Zeichen. Außerdem läßt sich der Grafikmodus durch den fehlenden Cursor erkennen. Der Cursor kann aber auch im Textmodus verschwinden, indem er auf Zeile 26 / Spalte 1 positioniert wird. Zeile 26 ist nämlich noch bis zur Spalte 48 vorhanden, nur kann diese Zeile nie angezeigt werden. Hierbei handelt es sich übrigens genau um die überzähligen 96 Bytes, von denen oben die Rede war. Es läßt sich also oft nur durch genaues Hinsehen oder mit der Alt-Methode feststellen, ob sich der Rechner gerade im Text- oder Grafikmodus befindet. Uns interessiert jedoch im folgenden nur noch der 80spaltige Textmodus.

BIOS-Interrupt 10H

Die Verwaltung des Bildschirmspeichers übernimmt das Betriebssystem über den Interrupt 10H (Video-Ein-/Ausgabe) aus dem ROM-BIOS. Da das ROM-BIOS fester Bestandteil des IBM-PC und Kompatibler ist, können auch andere Betriebssysteme die Bildschirmsteuerung nur über diesen INT 10H vornehmen. Insofern gilt das weitere zumindest theoretisch auch für andere Betriebssysteme auf dem PC. In dem Modus 2 (80 × 25

```
cseg    segment 'code'
        assume cs:cseg,ds:cseg
        org   100h          ;als .COM Datei konzipiert
;-------------------------------------------------------
;
;------ Anfang des spaeter residenten Teils ------------
;
anfang: jmp   biege         ;biegt int9 um und laedt resident
;
;****** INT 9 springt hierher ***************************
;                           ;-------------------------------
int9neu:pushf               ;Der Eintrag fuer den int9 in der
        db    09ah          ;Vektortabelle im HSP zeigt auf
int9alt:dw    0,0           ;diese Stelle, von wo ein wei-
        sti                 ;ter CALL zum alten int9 er-
        push  ax            ;folgt, der von dort mit einem
        push  bx            ;IRET hierher zurueckkommt
        push  cx            ;(Daher der pushf).
;                           ;-------------------------------
        mov   ah,1          ;Falls ueberhaupt kein Zeichen im
        int   16h           ;Tastaturpuffer abholbereit ist
        je    weiter        ;-> zurueck zum Betriebssystem
;                           ;-------------------------------
        or    al,al         ;Falls kein erweit.ASCII-Zeichen
        jne   weiter        ;-> zurueck zum Betriebssystem
;                           ;-------------------------------
        sub   ah,78h        ;Code auf 78h bis 7bh ueberpruefen
        mov   cl,ah         ; (entspricht Alt-1 bis 4).Falls
        and   ah,0fch       ; keine dieser Tasten gedrueckt
        jne   weiter        ; -> zurueck zum Betriebssystem
;                           ;-------------------------------
        mov   ah,15         ;nur die beiden Bildschirmmodi
        int   10h           ; 2/3 (Text 80x25 monochr/color)
        sub   al,2          ; erlauben 4 Bildschirmseiten
        and   al,0feh       ; die anderen werden ignoriert
        jne   weiter        ; ->zurueck zum Betriebssystem
;                           ;-------------------------------
        xor   ah,ah         ;Seitenumschaltung ist moeglich,
        int   16h           ;also Zeichen aus Puffer nehmen
;                           ;-------------------------------
        mov   al,cl         ;nun wird die gewuenschte Seite
        mov   ah,5          ;eingeschaltet (Seitennr. in cl)
```

Bild 1. Der Quellcode des Programms

Zeichen monochrom) und 3 (80 × 20 Zeichen in Farbe) erlaubt das ROM-BIOS, vier verschiedene Bildschirmseiten anzusprechen (BIOS Numerierung 0-3). Das sind vier aufeinanderfolgende Bereiche mit je 1000H Bytes, die zusammen den gesamten Speicher auf der Farbgrafikkarte ergeben, der übrigens immer fest bei Adresse B8000H (bis BB000H) beginnt. Über die Funktion 5 des INT 10H können diese Seiten eingeschaltet werden. Einschalten heißt, daß der entsprechende Speicherbereich auf dem Bildschirm angezeigt wird und fortan als „aktive Seite" gilt. Einige Funktionen des INT 10H nehmen Bezug auf diese aktive Seite, andere erlauben Aktionen auf einer frei wählbaren Seite, die nicht die aktive zu sein braucht. Beispielsweise gibt es über den INT 10H die Möglichkeit, Zeichen an die Cursorposition auf der gerade aktiven Seite auszugeben. Alle MS-DOS Betriebssystemprogramme gehen diesen Weg. Es können aber auch Zeichen auf eine anzugebende Seite an die dortige Cursorposition geschrieben werden. Jede Seite hat ihre eigene logische Cursorposition, die erst aktiviert wird, wenn die Seite angezeigt werden soll. Das Betriebssystem verwaltet allerdings ständig die Cursorpositionen aller Seiten.

Für die meisten Programmierer scheint die Bildschirmseite eher eine Last als eine Lust zu sein, sofern sie die Möglichkeit des Seitenwechsels überhaupt kennen. Es müßte sonst wenigstens einige Programme geben, die mehrere Seiten verwenden, um z.B. ein komplettes Infosystem zu ihrem Programm auf die restlichen Seiten zu legen, das blitzschnell zur Verfügung stünde und allen Anwendern das Leben erleichterte.

Viele Programme benutzen allerdings gar nicht den INT 10H für die Bildschirmausgabe, sondern die entsprechenden Funktionen des INT 21H, die keine Seitenwahl ermöglichen. Natürlich rufen die INT 21H Funktionen letztendlich die INT 10H Funktionen auf, jedoch nur solche, die die aktive Seite bedienen, und das ist gut so. Denn solange Programme auf die Cursorposition der aktiven Seite schreiben, spricht nichts dagegen, die aktive Seite zu wechseln, und die Ausgaben auf diese neue Seite schreiben zu lassen. Nichts anderes macht 4SEITEN.COM.

```
                int     10h             ;interne BIOS-Zaehlung ist  0-3
                mov     al,cl           ;zur besseren Orientierung  wird
                shl     al,1            ;der Cursor schliesslich auf die
                mov     cx,0607h        ;x-fache Groesse aufgeblasen,wo-
                sub     ch,al           ;bei x=Seitennummer gilt. Man-
                mov     ah,1            ;che Programme bestimmen die
                int     10h             ;Cursorgroesse jedoch  selbst
;                                       ;-------------------------------
weiter:         pop     cx              ;Alle Zeichen ausser  Alt-1 bis 4
                pop     bx              ;bleiben im Puffer. Andere Pro-
                pop     ax              ;gramme  koennen  sie auslesen;
                cli                     ;nichts ist verlorengegangen.
                iret                    ;Ruecksprung auf Systemebene.
;
;------ Ende des spaeter residenten Teils ----------------
;-------------------------------------------------------------------
biege:  push    es                      ;Der Interrupt-Vektor 9
        mov     ax,3509h                ;wird ueber die Fkt 35h
        int     21h                     ;des  int21h  geladen
        mov     word ptr int9alt,bx     ;und auf int9alt ge-
        mov     ax,es                   ;speichert,  wo  er
        mov     word ptr int9alt+2,ax;als Adr.fuer einen
        pop     es                      ;CALL FAR  dient.
;                                       ;-------------------------
        mov     dx,offset int9neu       ;Die  Adr. von  int9neu
        mov     ax,2509h                ;wird als  neuer int9-
        int     21h                     ;Vektor i.HSP gesetzt
;                                       ;-------------------------
        mov     dx,offset melde         ;Dem Anwender wird eine
        mov     ah,9                    ;Meldung / Bedienungs-
        int     21h                     ;anleitung ausgegeben
;                                       ;-------------------------
        mov     dx,offset biege         ;Speicher bis biege be-
        int     27h                     ;lassen,Rest freigeben
;-------------------------------------------------------------------
melde:  db      13,10,'       4SEITEN ist resident'
        db      13,10,'   Umschalten mit Alt-1 bis 4',13,10
        db      13,10,'(c) software peichl,heidelberg 1986'
        db      13,10,'$'
;-------------------------------------------------------------------
cseg    ends
        end     anfang
```

Was kann das Programm?

Vor dem Laden des Programms, welches mit DEBUG *(Bild 2)* eingegeben werden kann, sollte das System am besten neu gestartet werden. Auf dem Bildschirm erscheint danach die Meldung, daß 4SEITEN.COM resident geladen wurde und über die Tastenkombinationen Alt-1 bis Alt-4 die Seiten angewählt werden können. Das Betriebssystem schaltet immer Seite 1 (intern 0) ein. Mit Alt-2 kann auf Seite 2 umgeschaltet werden, wobei oben links ein etwas größerer Cursor zu sehen ist. Der Befehl DIR zeigt das Verzeichnis auf dieser Seite an. Mit Alt-3 kommen Sie auf die 3. Seite, wo der Cursor noch etwas größer ist. Hier können Sie sich das Inhaltsverzeichnis eines anderen Laufwerks ansehen. Alt-4 schaltet auf die 4. Seite, die durch die volle Cursorgröße gekennzeichnet ist. Mit dem Befehl TYPE CONFIG.SYS wird die jeweilige Systemkonfiguration auf Seite 4 gelistet. Jetzt kann mit Alt-1 bis Alt-4 beliebig zwischen den Seiten hin- und hergesprungen werden. Die Cursorgröße dient als Orientierung, auf welcher Seite man sich gerade befindet. Auf jeder Seite können weitere Befehle eingegeben werden, die dann auf der eingeschalteten Seite ausgeführt werden. Natürlich läßt sich mit dem Seitenwechsel nicht das Programm verlassen, mit dem auf dieser Seite gerade gearbeitet wird. Es wird nur die Ausgabe umgeleitet. Wir sind hier noch weit von einer Pseudo-Multitasking-Umgebung entfernt. Allerdings wäre ein Multitasking für maximal 4 Programme denkbar, die für ihre Ausgaben fest eine der 4 Seiten zugewiesen bekommen. Über den Seitenwechsel müßte dann das gerade verlassene Programm unterbrochen und das auf dem Bildschirm sichtbare aktiviert werden. Eine solche Umgebung sollte sich zumindest für einige Programme des MS-DOS ohne allzu großen Aufwand verwirklichen lassen.

Die Art der Programmierung von 4SEITEN läßt ein Umschalten zwischen den Seiten jederzeit zu, auch wenn der Rechner gerade keine Eingabe erwartet. Dies läßt sich leicht mit einer längeren Textdatei ausprobieren, die mittels des TYPE-Befehls gerade ausgegeben wird. Während die Ausgabe auf dem Bildschirm erscheint, kann auf eine andere Seite umgeschaltet werden, auf der dann die Ausgabe fortgesetzt wird. So lassen sich lange Texte über mehrere Seiten verteilen und danach kann in Ruhe, ohne die lästige Ctrl-S Unterbrechung, zurückgeblättert werden.

Eine Seite kann auch als Schmierzettel verwendet werden. Immer wenn etwas kurz notiert werden soll, geht man dazu auf Seite 4, macht die Notiz (ohne Enter-Taste), geht auf Seite 1 und drückt Enter. Dort erscheint dann die übliche Meldung „unknown...". Auf Seite 4 bleibt die Notiz jedoch sauber stehen, ohne alsbald am oberen Bildschirmrand zu verschwinden.
Der Experimentierfreude sind keine Grenzen gesetzt. Compilerfehlermeldungen können auf einer Seite verbleiben, während auf einer anderen diese Fehler behoben werden. Es kann nichts zerstört werden, da nur die Ausgabe betroffen ist. Wenn die Verwirrung auf einer Seite zu groß ist, braucht nicht neu gebootet zu werden. Es hilft der interne Betriebssystembefehl CLS, um die Seite gelöscht wieder zur Verfügung zu haben.

Ein Blick auf den Tastaturtreiber

Das eigentliche Problem besteht wie bei allen von außen aktivierbaren residenten Programmen darin, eine einfache Möglichkeit zu finden, das Programm jederzeit starten zu können. Das kann nur über den Tastatur-Hardware-Interrupt 9 geschehen. Jedesmal, wenn eine Taste gedrückt wird, aktiviert die Tastatur diesen Interrupt. Hinter dem INT 9 verbirgt sich übrigens nichts anderes als das Tastaturtreiberprogramm KEYBGR.COM. Die Tastatur funktioniert auch, wenn KEYBGR.COM nicht geladen wurde, allerdings mit einer anderen Tastenbelegung. Folglich muß das Programm KEYBGR.COM in ähnlicher Form noch einmal vorhanden sein, nämlich im ROM-BIOS, denn die Tastatur funktioniert bereits, bevor irgendeine Datei geladen wurde. Nationale Tastaturtreiber ersetzen tatsächlich den ursprünglichen INT 9 aus dem ROM-BIOS, lassen jedoch die kaum bekannte Möglichkeit zu, bei Bedarf auf den alten INT 9 zurückzugreifen. Nach der Eingabe von Ctrl-Alt-F1 steht die Tastatur in US-ASCII-Belegung, mit allen auf der deutschen Tastatur so schmerzlich vermißten Sonderzeichen, allerdings ohne Umlaute, zur Verfügung. Mit Ctrl-Alt-F2 kann wieder auf die nationale Belegung zurückgeschaltet werden.

Interruptvektoren werden umgebogen

Das Betriebssystem selbst ändert schon den INT 9, also liegt nichts näher, als ihn nocheinmal zu ändern, oder besser gesagt, umzuleiten und anzuzapfen. Die Initialisierung von 4SEITEN ändert den Interruptvektor 9 in der Interruptvektortabelle (Adresse 00024 bis 00028) auf einen Wert, der auf ein Programmstück in 4SEITEN zeigt. Dieser Programmabschnitt von 4SEITEN durchläuft zuerst den INT 9, was den Vorteil bietet, die Rohinformation, die der INT 9 erhält, bereits aufbereitet wiederzubekommen. Der INT 9 wandelt zum Beispiel Scan-Codes in ASCII-Codes und verarbeitet einige Tastenkombinationen wie Ctrl-Alt-Del selbständig. Die Information, die der INT 9 liefert, lässt sich leicht mit dem Software-Interrupt 16H abfragen. Wenn die gesuchten Tastenkombinationen (Alt-1 bis 4) gefunden wurden, werden eine Reihe von INT 10H Interrupts aufgerufen, die die Seite wechseln und den Cursor verändern.
Das Programm besteht aus einer Folge von verschiedenen Interruptaufrufen, zwei Werteprüfungen und einer kleinen Multiplikation. Mit dem Befehl PUSHF auf der Marke INT9NEU wird der INT 9 wie ein Unterprogramm aufgerufen. Es muß daher zuerst der Stack richtiggestellt werden, denn ein CALL schreibt nur 2 Halbworte auf den Stack, der aufgerufene INT 9 kehrt aber mit einem IRET zurück, und nimmt dabei 3 Halbworte vom Stack. Der zusätzliche Push-Befehl ist notwendig, um einen Stackfehler zu vermeiden. Der residente Teil von 4SEITEN wird auch wieder mit einem IRET verlassen, da er ja auch über einen Interrupt aktiviert wurde. Der Befehl DB 09AH definiert den OP-Code für den CALL FAR zu der darauffolgenden vollständigen Adresse.
Die Abänderung des INT 9 wird durch die beiden Funktionen 35H und 25H des INT 21H im Initialisierungsteil bewerkstelligt, der nur beim Laden des Programms durchlaufen wird und danach dem Betriebssystem wieder als freier Speicherplatz zur Verfügung steht (Freigabe über INT 27H).
Falls Programme laufen sollen, bei denen die Tastenkombinationen Alt-1 bis Alt-4 eine Bedeutung haben, braucht nur an der Stelle SUB AH,78H für 78H ein anderer Wert aus dem erweiterten ASCII-Code eingesetzt werden. Zu empfehlen sind die Werte 54H, 5EH oder 68H, weil sich dahinter jeweils fortlaufend die Funktionstasten F1 bis F4 verbergen, und zwar bei 54H in Verbindung mit der Shift-Taste, bei 5EH mit der Ctrl-Taste und bei 68H mit der Alt-Taste. Sollten auch die eben genannten Tasten nicht in Frage kommen, wäre eine größere Änderung nötig, weil 4SEITEN davon ausgeht, daß der Tastencode für die vier Seiten fortlaufend ist.
Die in 4SEITEN gewählte Methode zum Filtern der Tasten gewährleistet, daß keine Information der Tastatur verloren geht. Die gewünschten Tasten werden aus dem Puffer ausgelesen, alle übrigen werden weitergereicht, und stehen anderen Programmen uneingeschränkt zur Verfügung. In diesem Punkt scheinen einige residente Programme nachlässiger zu verfahren, was zu einem Tastaturpufferfehler führen kann. In diesem Zusammenhang sollen nun einige Konflikte angesprochen werden, die mit anderen Programmen entstehen können.

```
A>debug
-e 100  EB 45 90 9C 9A 00 00 00 00 FB 50 53 51 B4 01 CD
-e 110  16 74 2F 0A C0 75 2B 80 EC 78 8A CC 80 E4 FC 75
-e 120  21 B4 0F CD 10 2C 02 24 FE 75 17 32 E4 CD 16 8A
-e 130  C1 B4 05 CD 10 8A C1 D0 E0 B9 07 06 2A E8 B4 01
-e 140  CD 10 59 5B 58 FA CF 06 B8 09 35 CD 21 89 1E 05
-e 150  01 8C C0 A3 07 01 07 BA 03 01 B8 09 25 CD 21 BA
-e 160  6B 01 B4 09 CD 21 BA 47 01 CD 27 0D 0A 20 20 20
-e 170  20 20 20 20 34 53 45 49 54 45 4E 20 69 73 74 20
-e 180  72 65 73 69 64 65 6E 74 0D 0A 20 20 20 20 55 6D
-e 190  73 63 68 61 6C 74 65 6E 20 6D 69 74 20 41 6C 74
-e 1A0  2D 31 20 62 69 73 20 34 0D 0A 0D 0A 28 63 29 20
-e 1B0  73 6F 66 74 77 61 72 65 20 70 65 69 63 68 6C 2C
-e 1C0  68 65 69 64 65 6C 62 65 72 67 20 31 39 38 36 0D
-e 1D0  0A 24
-n 4seiten.com
-r cx
CX 0000
:d2
-w
Schreiben von 00D2 Bytes
-q
```

Bild 2. So können Sie mit Debug das Programm direkt eintippen

Probleme mit anderen speicherresidenten Programmen

Es ist klar, daß Programme, die nach 4SEITEN geladen werden und noch einmal den INT 9 verändern, potentiell Probleme bereiten. Übernimmt ein später geladenes Programm beispielsweise die INT 9 Verarbeitung vollständig selbst,

wird 4SEITEN gar nicht mehr durchlaufen. Einige Programme setzen sich auch an den Anfang des INT 9, lesen selbst den Tastaturkanal und durchlaufen danach ganz normal den INT 9, dem dann natürlich eine Tastaturinformation fehlt. Nun wird vielleicht auch klarer, warum die Anbieter residenter Programme empfehlen, ihr Programm immer als letztes zu laden, was bereits bei zwei Programmen sehr schwierig wird. Das zuletzt geladene Programm trifft die Interrupts immer so an, wie sie von ihm gesetzt wurden, und wird deshalb höchstwahrscheinlich laufen.
Allerdings traten bisher keine Konflikte mit anderen residenten Programmen auf, auch dann nicht, wenn 4SEITEN als erstes Programm geladen wurde. 4SEITEN ist nicht gegen mehrfaches Laden geschützt. Mehrere residente Versionen in Reihe behindern sich nicht.
Ein residentes Programm vor mehrfachem Laden zu schützen, oder mit einem zweites Laden die alten Interrupts wiederherstellen zu lassen, ist keinesfalls einfach, wenn nicht gar unmöglich, denn später geladene Programme könnten die Interrupts geändert haben.
Nun noch einige Bemerkungen zu den Erscheinungsbildern, die 4SEITEN auf dem Bildschirm hervorrufen kann. Überhaupt keine Probleme entstehen mit Programmen, die sich nach der aktiven Seite richten, wie beispielsweise das ganze MS-DOS. Am ärgerlichsten sind Programme, die den Bildschirmmodus ändern, da das BIOS dann immer alle 16K Bildschirmspeicher normiert. Damit sind natürlich alle Seiten gelöscht. Beim Umschalten in den Grafikmodus muß diese Normierung sein, weil ja die ganzen 16K gebraucht werden. Aber wenn Programme erst den Modus erfragen um ihn dann genauso wieder zu setzen, so ist das zwar eine einfache, aber keine weitsichtige Methode, den Bildschirm zu löschen.

Direktes Schreiben in den Bildschirmspeicher

Ein weiteres Problem entsteht, wenn Programme direkt in den Bildschirmspeicher auf Adresse B8000H schreiben. Dieses Vorgehen wird gemeinhin als schlechter Programmierstil bezeichnet. Die Ausgaben solcher Programme erscheinen natürlich nur auf Seite 1. Wenn dann zusätzlich auch noch Ausgaben an den Cursor gehen, der immer eine Variable der Seite ist, kann es passieren, daß ein Teil der Ausgaben auf Seite 1 und der Rest auf der aktiven Seite steht. Mit 4SEITEN besteht also auch die Möglichkeit der Qualitätsprüfung, denn gute Programme sollten die aktuelle Seite erfragen, und Ihre Ausgaben dorthin schicken, anstatt fest auf eine nicht mehr änderbare Seite zu schreiben.
Direktes Schreiben in den Bildschirmspeicher erzeugt eine sehr schnelle Ausgabe . Wenn ein Programm diese Anforderung erfüllen muß, empfiehlt es sich, je nach aktiver Seite auf die Adresse B8000H, B9000H, BA000H oder BB000H zu schreiben; das sind die Anfangsadressen der 4 Bildschirmseiten. Diese Vorgehensweise macht ein Programm flexibler und erfordert lediglich 5 Bytes mehr Code. Anstatt beispielsweise das Register BX direkt mit dem Segment B800H zu laden, müßten dafür vier neue Befehle eingegeben werden (*Tabelle*).
Mit etwas Geduld und Glück lassen sich manchmal auf diese Weise auch fertige Programme abändern, die danach auf jeder Seite lauffähig sind.

Tabelle: Direktes Schreiben auf den Bildschirm

```
mov ah, 0fh ;Bildschirmseite erfragen
int 10h     ;wird in BH zurückgegeben
add bh,b8h  ;Offset High-Byte Bildspeicher addieren
xor bl,bl   ;Low-Byte = 00H; neues Bildschirmsegment in BX
```

Spruch des Monats

„Es gibt einen weiteren Typ von Automaten: Solche, die nicht einfache Aktionen, sondern durchdachte Handlungen eines Menschen imitieren und den Menschen sogar ersetzen können."

Leonardo Torres Y Quevedo, Computerpionier, 1914.

Startadresse von Apple-Binärdateien

Bei DOS 3.3 kann man Binärdateien mit oder ohne Angabe der Anfangsadresse laden oder starten. Während an der Adresse $AA60 die Länge der Datei steht, fehlt die Möglichkeit, die Startadresse festzustellen, die in den ersten zwei Byte der Datei notiert ist.
Um die ORG-Adresse eines abgespeicherten Programms feststellen zu können, genügt ein einfacher Patch im DOS 3.3:

```
bisher:
A371: 20 7A A4          JSR $A47A

geändert:
A371: 20 __ __          JSR PATCH

neu:
__ __: 20 7A A4 PATCH   JSR $A47A
       8D F9 B7         STA $B7F9
       8C FA B7         STY $B7FA
       60               RTS
```

Man kann diesen 10 Byte langen Patch im Bereich $BA69...$BA95 bzw. $BCDF...$BCFF unterbringen, wenn darin noch genügend Platz vorhanden ist (DOS prüfen). Für „PATCH" ist die richtige Adresse einzusetzen. Mit INIT kann diese Änderung dauerhaft auf eine neue Diskette übertragen werden.
In $B7F9...$B7FA findet man nach dem Laden bzw. Starten einer Binärdatei deren ORG-Adresse, in $AA60...$AA61 die Länge, in $AA72...$AA73 die Ladeadresse.
In Applesoft kann man die Parameter so abfragen:

```
ORGADR =
PEEK(47097)+256*PEEK(47098)
LAENGE =
PEEK(43616)+256*PEEK(43617)
LODADR =
PEEK(43634)+256*PEEK(43635)
```

Wolfdieterich Müller

Matthias Köfferlein

Disk-Editor für CP/M-68k

Das hier beschriebene Programm ermöglicht es, einzelne Sektoren einer Diskette in den Speicher zu laden und auszugeben, Änderungen am Inhalt dieser Sektoren vorzunehmen und den geänderten Inhalt auch wieder auf die Diskette zurückzuschreiben. Er gibt außerdem die Disketten-Parameter an, also die Zahl der reservierten Systemspuren oder der Spuren insgesamt.

Das Programm besteht aus zwei Teilen. Der größte Teil des Programms ist in C geschrieben (*Bild 1*). Lediglich die Einbindung des Programms in das Betriebssystem erfolgt über Assemblerroutinen, die den anderen Teil bilden (*Bild 2*). Beide Programmteile müssen mit einem Editor (z. B. ED.68k) eingegeben werden. Dabei können die Kommentare natürlich weggelassen werden. Nachdem man beide Teile eingegeben hat, müssen sie übersetzt werden. Hat man z.B. das C-Programm unter dem Namen DISKED.C abgespeichert, so erfolgt das Übersetzen dieses Programmteils mit der Sequenz

C DISKED

C.SUB ist die Submit-Datei, die bei CP/M-68k zum C-Compiler mitgeliefert wird. Der Assemblerteil wird, wenn er unter dem Namen CPMC.S abgespeichert wurde, mit der Sequenz

AS68 -L CPMC.S

übersetzt. Danach müssen noch beide Teile mit dem Linker gebunden werden. Dies geschieht mit dem Kommando

LO68 -R -U_NOFLOAT -O DISKED.REL S.O DISKED.O CPMC.O CLIB

Man erhält damit die direkt ausführbare relokatible Datei DISKED.REL.

Anpassung

Um das Programm an die eigene Hardware anzupassen, müssen eventuell die Steuercodes für den Bildschirm geändert werden. Die Steuercodes sind alle am Anfang des Programms als Makros mit der #define-Anweisung des C-Prozessors definiert. Auch die Funktion für 'Cursor direkt setzen' ist direkt als Funktion per #define-Anweisung definiert. In der abgedruckten Version arbeitet das Programm mit den Steuerzeichen des mc-68000-Computers.
Nachdem das Programm gestartet wurde, wird zuerst der Bildschirm gelöscht und die Bildschirm-Maske ausgegeben. Danach stellt das Programm fest, welches Laufwerk aktiv war, als es gestartet wurde, liest Sektor 0, Spur 0 von der Diskette in den Speicher und gibt den Inhalt dieses Bereichs aus. Gleichzeitig liest das Programm die Parameter des angesprochenen Laufwerks ein und zeigt diese im rechten unteren Bildschirmviertel an.

Die Kommandos

Nach der Initialisierungsphase wartet das Programm auf ein Kommando von der Tastatur. Der Speicherinhalt wird als ein Feld von 16 mal 8 Bytes dargestellt; mit den Cursortasten kann nun der Cursor innerhalb der Darstellung verscho-

```
#define CLS 12                          /* Definitionen: Schirm loeschen */
#define UP 30                           /*   Cursor nach oben */
#define DOWN 31                         /*   Cursor nach unten */
#define LEFT 29                         /*   Cursor nach links */
#define RIGHT 28                        /*   Cursor nach rechts */
#define ESC 27                          /*   Escape */
#define BS 8                            /*   Backspace */
#define CR 13                           /*   Carriage Return */
#define HOME 11                         /*   Code fuer Taste: Home */
#define END 17                          /*                    Linke u. Ecke */

#define scan() bios(3, 0)               /* Keyboard abfragen: BIOS Fkt. #3 */
#define goxy(xp,yp) printf("\033y%c%c",(xp)+32,(yp)+32)
                                        /* Sequenz fuer Cursor absolut setzen */
#define clrel() printf("\033k")         /* Sequenz fuer Zeilenende loeschen */

char buffer[128];                       /* Puffer fuer Sektor */
int sect = 0;                           /* Sektornummer (physikalisch) */
int track = 0;                          /* Tracknummer */
int cursx = 0;                          /* Position des Cursors */
int cursy = 0;
int drive;                              /* Nummer des Drives (0:A usw.) */
char hexl[] = "0123456789abcdef";       /* ASCII-Codes der Hex-Zahlen */
char hexu[] = "0123456789ABCDEF";       /* Auch als upper-case Buchstaben */

struct dpb {                            /* Disk parameter block */
        int spt;                        /* Zahl der records (128 Byte)/track */
        char bsh;                       /* 3: 1024, 4: 2048, 5: 4096 usw ... */
        char r1;                        /* 3 unbenuetzte Bytes */
        char r2;
        char r3;
        int dsm;                        /* Diskgroesse in Blocks minus 1 */
        int drm;                        /* Directory-Eintraege minus 1 */
        int r4;                         /* 2 unbenuetzte Worte */
        int r5;
        int off;                        /* Zahl der reservierten Tracks */
};
struct dph {                            /* Disk Parameter Header */
        char *xlt;                      /* Addresse der translation table */
        int r1;                         /* 5 unebenuetzte Woerter */
        int r2;
        int r3;
        long r4;
        struct dpb *diskp;              /* Addresse des DPB */
};
char *xtable;                           /* Zeiger auf translation table */
struct dpb *diskpar;                    /* diskpar: Enthaelt Diskparameter */
int tracks;                             /* Anzahl der Tracks */

extern long bdos();                     /* Funkion zur Einbindung des BDOS */
extern long bios();                     /* Funktionen zur Einbindung des BIOS */
extern long biosl();                    /* Fuer Longword-Parameter */
extern int sectxlt();                   /* Sector-Translation */
```

Bild 1. Der Disk-Editor ist überwiegend in C geschrieben

```
main()                          /* Hauptprogramm */
{
char c = '\0';

drive = bdos(25, 0);            /* BDOS Fkt. #25 (Get current disk) */
title();                        /* Titelbild */
login();                        /* Diskparameter holen */
reads();                        /* Sektor 0,0 lesen */
display();                      /* Pufferinhalt ausgeben */
while(c != ESC) {               /* ESC beendet Hauptprogramm */
        goxy(cursx * 3 + 5, cursy + 5);
                                /* Cursor setzen */
        c = scan();             /* Keyboard abfragen */
        switch (c) {
        case DOWN:
                if (++cursy < 8)        /* Cursor nach unten bewegen */
                        break;
        case UP:
                if (cursy)              /* Cursor nach oben bewegen */
                        cursy--;
                break;
        case 'e':
        case 'E':
                editby();               /* E: Byte aendern */
                displn(cursy);          /* Zeile ausgeben und weiter */
        case RIGHT:
                if (++cursx < 16)       /* Cursor nach rechts bewegen */
                        break;
                if (++cursy < 8) {
                        cursx = 0;
                        break;
                }
                cursy--;                /* Kein break: Cursor links */
        case LEFT:
                if (cursx) {            /* Cursor nach links bewegen */
                        cursx--;
                        break;
                }
                if (cursy) {
                        cursy--;
                        cursx = 15;
                }
                break;
        case 's':
        case 'S':
                setts();                /* S: Track und Sektor setzen */
                break;
        case 'r':
        case 'R':
                reads();                /* R: Sektor lesen */
                display();              /* Pufferinhalt ausgeben */
                break;
        case 'w':
        case 'W':
                if (ask())              /* Sicherheitsabfrage fuer W: */
                        writes();       /* Sektor schreiben */
                break;
        case ESC:
                if (!ask())             /* Sicherheitsabfrage bei ESC */
                        c = 0;          /* Bei Nein: ESC entschaerfen */
                break;
        case HOME:
                cursy = cursx = 0;      /* HOME: Cursor n. rechts o. */
                break;
        case END:
                cursy = 7;              /* END: Cursor n. links u. */
                cursx = 0;
                break;
        case 'l':
        case 'L':
                seldsk();               /* L: Laufwerk anwaehlen */
                dispts();               /* Statuszeile aktualisieren */
                break;
        }
        putchar(CLS);                   /* Zurueck ins System */
        bdos(0, 0);                     /* BDOS Fkt. #0 (System reset) */
}

title()                         /* Titelbild ausgeben */
{
        int i;

        putchar(CLS);
        printhl();
        puts("\t\t     SEKTOR <T:  ,S:  > LAUFWERK <  : >");
        puts("\n      x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 xA xB xC xD xE xF\n");
        puts("\n\n\n\n\n\n\n\n\n\n\n\t    <S> TRACK UND SEKTOR SETZEN");
        puts("\t     <R> SEKTOR LESEN\t\t  RECORDS PRO TRACK <  >");
        puts("\t     <W> SEKTOR SCHREIBEN\t\t  BLOCKGROESSE <    >");
        puts("\t     <E> ANGEWAEHLTES BYTE AENDERN\tDISKETTENGROESSE <    >");
        puts("\t     <L> LAUFWERK ANWAEHLEN\t  DIRECTORY-EINTRAEGE <    >");
        puts("\t  <ESC> EDITOR VERLASSEN\t  RESERVIERTE TRACKS <  >");
        puts(" <CURSORTASTEN> CURSOR BEWEGEN\t\t\t\t  TRACKS <    >");
}

display()                       /* Pufferinhalt als Hexdump ausgeben */
{
        int i;

        for (i = 0; i < 8; i++)         /* 8 Zeilen Hexdump ausgeben */
                displn(i);
        dispts();                       /* Statuszeile aktualisieren */
}

dispts()                        /* Statuszeile aktualisieren */
{
        goxy(28, 1);                    /* Track & Sektor ausgeben */
        printf("<T:%02x,S:%02x>", track, sect);
                                        /* T & S als zweistellige Hexzahlen */
        goxy(49, 1);                    /* Drivenummer und Drivecode ausgeben */
        printf("<%02x:%c>", drive, drive + 'A');
}

displn(line)                    /* Eine Zeile Hexdump ausgeben */
int line;                       /* Zeile: 0..7 */
{
        int i;
        char *p, ascii[17];             /* ascii[]: Puffer fuer ASCII-Dump */

        goxy(0, line + 5);              /* Cursor positionieren */
        printf("%cx   ",line + 48);     /* Zeilennummer ausgeben (0x .. 7x) */
        p = buffer + line * 16;         /* p: Zeiger auf auszugebendes Byte */
        for (i = 0; i < 16; i++) {
                printf("%02x ", (int)*p & 0xff);
                                        /* Byte als 2-st. Hexzahl ausgeben */
                ascii[i] = (*p & 0x60) ? *p & 0x7f : '.';
                                        /* ASCII-Dump-String aufbauen: */
                                        /* ' ' falls Zeichen kein ASCII-Code */
                p++;
        }
        ascii[16] = '\0';               /* ASCII-Dump-String abschliessen */
        printf("  %s", ascii);          /* Und ausgeben */
}

readby(def)                     /* Byte als 2-stellige Hex-Zahl eingeben */
int def;                        /* def: default-Wert */
{
        int c, i;
        int pos = 0;                    /* pos: Eingabeposition (0 .. 2) */
        int by = 0;                     /* by: Puffer fuer Byte */

        printf("  %c%c", LEFT, LEFT);   /* Feld loeschen */
        while ((c = scan()) != CR) {    /* CR beendet Eingabe */
                for (i = 0; i < 16; i++) {
                        if (hexu[i] == c || hexl[i] == c) {
```

```
                                        /* Falls Hex-Ziffer: */
                        if (pos != 2) {
                                by = 16 * by + i;
                                        /* Nibble abspeichern */
                                pos++;
                                putchar(c);
                        }
                }
        }
        if.(c == BS) {
                if (pos) {              /* Backspace: Nibble loeschen */
                        putchar('\b');
                        by /= 16;
                        pos--;
                }
        }
        if (c == ESC) {
                while (pos) {           /* ESC: default-Werte holen */
                        pos--;
                        putchar(LEFT);
                }
                printf("%02x", by = def);
                                        /* Und als Byte abspeichern */
                pos = 2;
        }
    }
    return(by);                         /* Byte zurueckgeben */
}

editby()                                /* Byte editieren */
{
    char *p;                            /* Addresse des aktuellen Bytes */

    p = buffer + cursx + cursy * 16;
    *p = readby((int)*p & 0xff);        /* Byte editieren */
}

setts()                                 /* Track und Sektor setzen */
{
    goxy(31, 1);                        /* Cursor aus Track-Feld */
    track = readby(track) % tracks;     /* Track eingeben */
    dispts();                           /* Eingegebenen Wert anzeigen */
    goxy(36, 1);                        /* Dasselbe mit der Sektornummer */
    sect = readby(sect) % (diskpar -> spt);
    dispts();
}

reads()                                 /* Sektor lesen */
{
    int stat;                           /* stat: Status beim Lesen */

    setsek();                           /* Sektorparameter setzen */
    stat = bios(13, 0);                 /* BIOS Fkt. #13 (Read sector) */
    if (stat) {                         /* Falls stat == 1: Fehler */
            putchar(HOME);
            clrel();
            printf("READ ERROR AUF <T:%02x,S:%02x>", track, sect);
    }
    return(stat);
}

setsek()                                /* Sektorparameter setzen */
{
    bios(9, drive);                     /* BIOS Fkt. #9 (Select disk drive) */
    bios(10, track);                    /* BIOS Fkt. #10 (Set track number) */
    bios(11, sectxlt(xtable, sect));
                                        /* BIOS Fkt. #11 (Set sector number) */
    biosl(12, buffer);                  /* BIOS Fkt. #12 (Set DMA address) */
}

writes()                                /* Sektor schreiben */
{
    int stat;                           /* stat: Status beim Schreiben */

    setsek();                           /* Sektorparameter setzen */
    stat = bios(14, 1);                 /* BIOS Fkt. #14 (Write sector) */
                                        /* Directory-Sektor schreiben (1) */
    if (stat) {                         /* stat == 1: Schreibfehler */
            putchar(HOME);
            clrel();
            printf("WRITE ERROR AUF <T:%02x,S:%02x>", track, sect);
    }
}

ask()                                   /* Sicherheitsabfrage */
{
    int c;

    putchar(HOME);                      /* Ueberschrift loeschen */
    clrel();
    printf("SIND SIE SICHER <J/N> ? ");
    c = scan();                         /* Keyboard abfragen */
    printhl();                          /* Ueberschrift wieder schreiben */
    return(c == 'j' || c == 'J');
                                        /* Rueckgabe: 1: JA, 0: Sonst */
}

printhl()                               /* Ueberschrift ausgeben */
{
    putchar(HOME);
    puts("DISKEDITOR CP/M 68k, MATTHIAS KOEFFERLEIN 1986");
}

seldsk()                                /* Drive auswaehlen */
{
    goxy(50, 1);                        /* Cursor auf Feld positionieren */
    drive = readby(drive) & 0xf;        /* Drivenummer einlesen (0 .. 15) */
    login();
}

login()                                 /* Login fuer aktuelle Disk */
{
    int stat;
    struct dph *p;                      /* Zeiger auf DPH */

    track = 0;                          /* Track und Sektor auf 0 setzen */
    sect = 0;
    dispts();                           /* Und anzeigen */
    p = bios(9, drive);                 /* BIOS Fkt. #9 (Select disk drive) */
    xtable = p -> xlt;                  /* Addresse der translation table */
    diskpar = p -> diskp;               /* Addresse des DPB */
    dispdpb();                          /* DPB anzeigen */
}

dispdpb()                               /* DPB anzeigen */
{
    int bls;                            /* bls: Blockgroesse */

    bls = 128 << (diskpar -> bsh);      /* Blockgroesse berechnen */
    goxy(66, 17);                       /* SPT ausgeben */
    printf("%02x", diskpar -> spt);
    goxy(66, 18);                       /* BSH ausgeben */
    printf("%04x", bls);
    goxy(66, 19);                       /* DSM ausgeben (in kBytes) */
    printf("%04x", ((diskpar -> dsm) + 1) * (bls >> 10));
    goxy(66, 20);                       /* DRM ausgeben (Dir.-entries) */
    printf("%04x", (diskpar -> drm) + 1);
    goxy(66, 21);                       /* OFF ausgeben (resvd tracks) */
    printf("%02x", diskpar -> off);
    tracks = (diskpar -> off) +
            ((long)((diskpar -> dsm) + 1) * bls) / (128 * (diskpar -> spt));
                                        /* Anzahl der Tracks berechnen */
    goxy(66, 22);                       /* Und ausgeben */
    printf("%04x", tracks);
}
```

ben werden. Alle Kommandos werden zur Erinnerung im linken unteren Bildschirmviertel angezeigt. Als Kommandos stehen außer den Cursortasten zur Verfügung:

Byte ändern (E)
Mit diesem Kommando kann das Byte, auf das der Cursor zeigt, verändert werden. Die Eingabe wird hexadezimal erwartet. Wird keine Ziffer eingegeben, so wird der Wert 0 angenommen. Mit der ESC-Taste kann der alte Wert in das Eingabefeld übernommen werden. Return beendet die Eingabe.

Spur-/Sektornummer setzen (S)
Damit kann die Spur- und Sektornummer für einen folgenden Schreib- bzw. Lesezugriff gesetzt werden. Zuerst wird die Eingabe der Spurnummer erwartet. Dabei gelten die selben Regeln, wie sie eben beim E-Befehl erläutert wurden. Nach Druck auf die CR-Taste muß die Sektornummer eingegeben werden, die Eingabe erfolgt wie bei der Spurnummer. Werden bei der Eingabe die zulässigen Grenzen für Spur- bzw. Sektornummer überschritten, so werden diese modulo der jeweils höchsten zulässigen Nummer genommen.

Die Sektornummer entspricht der logischen Sektornummer auf der Disk. Damit entfallen lästige Berechnungen, wenn bei dem verwendeten Format die Sektoren nicht hintereinander auf der Disk liegen, sondern mit Zwischenräumen auf der Diskette untergebracht sind (Sector translation).

Sektor lesen (R)
Der Sektor, dessen Adresse mit dem S-Kommando eingegeben wurde, wird in den Speicher eingelesen. Ein eventuell auftretender Lesefehler wird in der obersten Zeile signalisiert.

Sektor schreiben (W)
Damit diese Funktion nicht versehentlich ausgelöst werden kann, ist zusätzlich eine Sicherheitsabfrage eingebaut, die mit J beantwortet werden muß.

Trotzdem hindert das Programm den Benutzer nicht daran, wertvolle Daten zu zerstören. Deshalb sollte man den Befehl äußerst vorsichtig benutzen.

Laufwerk ändern (L)
Mit diesem Kommando kann das aktuelle Laufwerk gewechselt werden. Die Eingabe der Laufwerksnummer erfolgt hexadezimal. Für Laufwerk A muß 0 eingegeben werden, für B 1, C 2 usw. Die Laufwerksnummer wird immer modulo 16 genommen. Der Buchstabe des Laufwerks wird hinter der Nummer angegeben. Für die Eingabe gelten dieselben Regeln, wie für das S- oder E-Kommando. Nachdem die Laufwerksnummer eingegeben wurde, liest das Programm die Parameter des angewählten Laufwerks ein und stellt sie im rechten unteren Bildschirmviertel dar. Dabei sind alle Angaben hexadezimal. Die Blockgröße wird im Bytes, die Diskettengröße in KBytes angegeben. Weiterhin wird die Anzahl der Spuren, der Sektoren pro Spur (ein Sektor entspricht 128 Bytes), die Zahl der Einträge im Inhaltsverzeichnis und die Anzahl der reservierten Systemspuren angezeigt.

Programm beenden (ESC)
Mit ESC kehrt das Programm ins Betriebsystem zurück. Vorher wird jedoch noch eine Sicherheitsabfrage durchgeführt, die mit J beantwortet werden muß.

Das Inhaltsverzeichnis

Bild 3 zeigt eine Hardcopy des Bildschirms nach dem Start des Programms. Das Diskettenformat auf diesem Bild ist das Format, das der mc-68000-Computer benützt. Bei diesem besteht eine Spur aus 40 (Hexadezimal $28) Sektoren. Vier Systemspuren sind für Ladeprogramm und Betriebsystem reserviert. Beim mc-68000-Computer liegt das CP/M jedoch im EPROM, so daß sich auf diesen Spu-

```
* ------------------------------------------------------------------------
*
*       Einbindung der BDOS- und BIOS-Funktionen in das C-Programm:
*
*               _sectxtl:       Sector translation
*               _bdos:          BDOS-Aufruf
*               _bios:          BIOS-Aufruf
*               _biosl:         BIOS-Aufruf mit Langwort-Argument
*
* ------------------------------------------------------------------------
*
*       Beim Aufruf der Funktionen wird der Wert im D0-Register als
*       Funktionswert zurueckgegeben.
*       Defaultmaessig wird ein Word-Parameter zurueckgegeben. Werden die
*       Funktionen mit
*
*               extern long bios(), biosl(), bdos();
*
*       definiert, so wird D0 als Langwort von C uebernommen.
*       Aufruf der Funktionen:
*
*               ret = biosl((int)function#, (long)arg);
*               ret = bios((int)function#, (int)arg);
*               ret = bdos((int)function#, (int)arg);
*               xsect = sectxtl((char *)table, (int)sect);
*
* ------------------------------------------------------------------------
        .globl _biosl                   * Als globale Funktionen definieren
        .globl _bios
        .globl _sectxlt
        .globl _bdos
* ------------------------------------------------------------------------
        .text

_bios:
        move.w 4(sp),d0                 * Funktionsnummer vom Stack holen
        move.w 6(sp),d1                 * Argument vom Stack holen
        clr.w d2                        * D2 loeschen
        trap #3                         * BIOS-Aufruf
        rts                             * Zurueck ins C-Programm

_biosl
        move.w 4(sp),d0                 * Dasselbe mit Long-Parameter
        move.l 6(sp),d1
        clr.w d2
        trap #3
        rts

_bdos:
        move.w 4(sp),d0                 * Einbindung des BDOS in C
        move.w 6(sp),d1                 * (wie BIOS)
        trap #2
        rts

_sectxlt:                               * Sector translation
        move.l 4(sp),d2                 * Addresse der Tabelle vom Stack
        move.w 8(sp),d1                 * Sektornummer vom Stack
        move.w #16,d0                   * BIOS Fkt. #16 (Sector translate)
        trap #3                         * BIOS-Aufruf
        rts
* ------------------------------------------------------------------------
        end
```

Bild 2. Diese Assemblerprogramme stellen die Verbindung zwischen dem C-Programm und dem Betriebssystem her

```
DISKEDITOR CP/M 68k, MATTHIAS KOEFFERLEIN 1986
                     SEKTOR <T:04,S:04> LAUFWERK <00:A>

      x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 xA xB xC xD xE xF

0x    00 4A 4F 59 20 20 20 20 20 4F 20 20 00 00 00 06    .JOY      O  ....
1x    73 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    s...............
2x    00 43 50 36 38 20 20 20 20 36 38 4B 00 00 00 80    .CP68     68K....
3x    5C 00 5D 00 5E 00 5F 00 60 00 61 00 62 00 63 00    \.].^._.`.a.b.c.
4x    00 43 50 36 38 20 20 20 20 36 38 4B 01 00 00 47    .CP68     68K...G
5x    64 00 65 00 66 00 67 00 68 00 00 00 00 00 00 00    d.e.f.g.h.......
6x    00 53 54 44 49 4F 20 20 20 48 20 20 00 00 00 1D    .STDIO    H    ....
7x    19 00 1A 00 00 00 00 00 00 00 00 00 00 00 00 00    ................


          <S> TRACK UND SEKTOR SETZEN
          <R> SEKTOR LESEN                    RECORDS PRO TRACK <28>
          <W> SEKTOR SCHREIBEN                    BLOCKGROESSE <0800>
          <E> ANGEWAEHLTES BYTE AENDERN      DISKETTENGROESSE <030C>
          <L> LAUFWERK ANWAEHLEN           DIRECTORY-EINTRAEGE <0100>
        <ESC> EDITOR VERLASSEN             RESERVIERTE TRACKS <04>
<CURSORTASTEN> CURSOR BEWEGEN                          TRACKS <00A0>
```

Bild 3. So präsentiert sich der Disk-Editor dem Benutzer auf dem Bildschirm

ren nur Konfigurations-Informationen befinden. Die Systemspuren liegen auf Spur 0...3, in Spur 4 schließt sich das Inhaltsverzeichnis an. Es umfaßt bei diesem Format 256 ($100) Einträge. Jeder Eintrag besteht aus 32 Bytes. Das gesamte Inhaltsverzeichnis umfaßt also 8192 Bytes, dies entspricht, bei der verwendeten Blockgröße von 16 Sektoren und somit 2048 (= $800) Bytes pro Block, vier Blöcken. Die Blöcke sind sequentiell angeordnet, d. h. Block 0 beginnt auf Spur 4, Sektor 0. Block 1 beginnt auf Spur 4, Sektor 16, und Block 2 auf Spur 4, Sektor 32. Da dieser Block über das Ende der Spur hinausreicht, werden auch noch die ersten 8 Sektoren der nachfolgenden Spur zu diesem Block hinzugenommen, so daß Block 3 bei Spur 5, Sektor 8 beginnt. Da das Inhaltsverzeichnis 4 Blöcke umfaßt, beginnen die eigentlichen Daten bei Block 4, also bei Spur 5, Sektor 24. Das Bild zeigt den Inhalt der Spur 4, Sektor 4, also einen Ausschnitt aus dem Inhaltsverzeichnis. Dies besteht aus Einträgen von je 32 Byte Länge. Jeder Eintrag adressiert 8 Blöcke, so daß ein Eintrag 8 × 2048 = 16384 Bytes adressieren kann. Ist eine Datei größer, so nimmt sie entsprechend viele Einträge in Anspruch (Extensions). Jeder Eintrag im Inhaltsverzeichnis entspricht funktionsmäßig den ersten 32 Byte eines FCB (File control block), und ist folgendermaßen aufgebaut:

Byte 0: Ist dieses Byte mit $E5 beschrieben, so gilt der Eintrag als gelöscht.

Durch Überschreiben mit $00 können also gelöschte Dateien wieder hergestellt werden. Dabei ist zu beachten, daß eine Datei auch aus mehreren Extensions bestehen kann, die alle mit $00 gekennzeichnet werden müssen. Diese Methode funktioniert außerdem auch nur dann einwandfrei, wenn nach dem Löschen keine Schreib-Operationen auf der Diskette durchgeführt wurden. Hat dieses Byte einen Wert ungleich $E5, so gibt es die Usernummer an, unter der die Datei angesprochen werden kann. Dieses Byte unterscheidet sich in der Bedeutung somit von der Bedeutung im FCB, wo es die Drivenummer angibt.

Byte 1...11: Diese Bytes enthalten den Dateinamen: Die ersten acht Bytes enthalten den Namen, die folgenden drei Bytes die Kennung. Unbenützte Bytes werden mit $20 (Leerzeichen) aufgefüllt. Die drei Bytes der Kennung enthalten zusätzlich Informationen über den Status der Datei in den höchstwertigen Bits. Nähere Informationen darüber gibt [1].

Byte 12: Dieses Byte enthält die Extensionnummer. $00 bedeutet, daß dieser Eintrag der erste Eintrag der Datei ist.

Byte 13...14: Diese Bytes haben keine Bedeutung.

Byte 15: Dieses Byte gibt an, wieviele Sektoren (128 Bytes je Sektor) durch diesen Eintrag adressiert werden. Ist der Eintrag voll aufgefüllt, so enthält dieses Byte bei einer Blockgröße von 16 Sektoren, also den Wert 16 × 8 = 128 ($80).

Byte 16...31: Diese Bytes stellen eine Liste dar, in der die Blöcke angegeben werden, die durch diesen Eintrag belegt werden. Dabei müssen zwei grundsätzliche Fälle betrachtet werden. Sind auf der Diskette weniger als 256 Blöcke vorhanden, so wird jedem Block ein Byte dieser Liste zugeordnet, wobei nur die Bytes 16...23 benützt werden. Sind auf der Diskette mehr als 256 Blöcke vorhanden, so werden jedem Block der Liste 2 Bytes zugeordnet. Das erste der beiden enthält das Low-Byte der Blocknummer, das zweite Byte das dazugehörige High-Byte.
Auf eine Möglichkeit der Manipulation wurde schon hingewiesen. Man kann mit dem Diskeditor natürlich z. B. den Status einer Datei nachträglich ändern oder Blöcke aus der Datei löschen. Man sollte diese Manipulation nur an Sicherheitskopien durchführen, da zerstörte Daten sonst kaum wiederbeschafft werden können.

Literatur

[1] Digital Research: CP/M 68k Operating System Programmer's Guide, 1983.
[2] Digital Research: The C-Language Programming Guide, 1983.
[3] Digital Research: CP/M 68k Operating System Guide, 1983.
[4] *Kernighan, Brian W.:* Programmieren in C, Hanser-Verlag, Wien; 1983.

Programmieren in Assembler

Während der Entwicklung eines Programms in Assembler oder einer Hochsprache muß unzählige Male der Editor gestartet, die Quelldatei geladen, die eingegebenen Daten abgespeichert, der Assembler/Compiler und der Linker gestartet werden. Diese mühseligen und zeitraubenden Arbeiten können mit dem von Lauer & Wallwitz entwickelten SAL-System schneller durchgeführt werden. SAL (Structured and Assembly Language) bietet eine komfortable Benutzeroberfläche für den Programmierer mit den Komponenten:

- eingebauter Editor
- interaktive Schnittstelle zu Assembler, Compiler, Linker, Debugger und Library Manager
- komfortable DOS-Schnittstelle.

Da ähnliche Steueranweisungen wie in Turbo-Pascal verwendet werden können, ist eine einfachere Programmierung in Assembler möglich. Repeat..Until, IF..Then..Else, Case, While..Do werden von SAL verstanden und in MASM-Quelltext übersetzt. Der Assembler-Programmierer kann damit Programme in kürzerer Zeit erstellen. *Dieter Strauß*

Rudolf Bremer

Turbo-Pascal bedient Hercules-Karte

Die Hercules-Grafikkarte unterstützen heute viele professionelle Programme. Leider gilt das nicht für Turbo-Pascal. Aber das kann man ändern. Unser Beitrag zeigt, was zu tun ist.

Die im folgenden vorgestellten Prozeduren und Funktionen (*Bild*) können als Include-Datei in eigene Programme eingebunden werden. Es handelt sich nur um grundlegende Routinen, da Algorithmen für Kreise usw. schon mehrfach in Zeitschriften und Büchern erschienen sind.
Alle Variablen sind lokal deklariert. Es besteht natürlich auch die Möglichkeit, einen Teil global im Deklarationsteil des jeweiligen Programms vorzugeben oder eine separate Include-Datei mit allen Deklarationen anzulegen.
Für die MS-DOS-Version von Turbo-Pascal stehen auch die Routinen Sound-(Freq) und NoSound zur Verfügung. Sie müssen für die PC-DOS-Version entfernt werden.
Alle Koordinatenangaben müssen folgende Bedingung erfüllen:
$0 \leqq X \leqq 719$
$0 \leqq Y \leqq 347$
In den Routinen ist keine Bereichsüberprüfung vorgesehen. Das aufrufende Programm muß also sicherstellen, daß die übergebenen Parameter im erlaubten Bereich liegen.

Die Routinen im einzelnen:

SoundKanalON
Einschalten des Lautsprechers.

NoSound
Ausschalten des Lautsprechers.

Sound(Freq:Integer)
Festlegen der Frequenz. Der Wert, mit dem der „Counter" geladen wird, resultiert aus folgendem Ausdruck:
Counter := 1.19318E6 / Freq
Als niedrigste Frequenz wird 37 Hz akzeptiert.

Beep(Anzahl:Integer)
Ausgabe von N Pieptönen, z. B. Fehlermeldungen.

HerculesMode(Option:Boolean)
Umschalten zwischen Grafik-(Full) und Text-Modus (Diag).
Option = true aktiviert den Grafik-Modus. Bevor man das Programm beendet, muß man den Grafik-Modus unbedingt ausschalten.

ByteOffset(x,y:Integer):Integer
Diese Funktion stellt die Offset-Adresse des Bytes fest, mit der die Koordinate x,y angesprochen wird. Die Segment-Adresse für die zweite Seite ist B800h.

BitOffset(x:Integer):Integer
Mit dieser Routine wird die Bitposition der X-Koordinate im Offset-Byte festgestellt.

TestDot(x,y:Integer):Boolean
Ob ein Pixel an der vorgegebenen Koordinate gesetzt ist, läßt sich mit dieser Routine feststellen.
True: Punkt gesetzt.
False: Punkt nicht gesetzt.

SetResDot(x,y:Integer;SetRes:Boolean)
Setzen und Löschen eines Bildpunktes erfolgt durch die SetResDot-Prozedur.

SetResByte(x,y:Integer;SetRes:Boolean)
Setzen und Löschen von jeweils acht Pixels.

FillSegment(x,y,AnzX,AnzY:Integer; SetRes:Boolean)
Löschen bzw. Setzen eines Bildschirmbereiches.
Da diese Prozedur byteweise arbeitet, sollten die X-Parameter ein ganzzahliges Vielfaches von 8 sein.

SwitchDot(x,y:Integer)
Um z. B. einen Cursor zu Erzeugen, ist es nötig, Punkte an bestimmten Positionen zu invertieren. Diese Prozedur ist dazu geeignet.

ClearHercules
Ein schnelles Löschen des Grafikbereiches gewährleistet diese Routine.

```
(*-------------------------------------------------------------------------*)
(* Routinen zur Lautsprecheransteuerung fuer die MS-DOS Version           *)
(*-------------------------------------------------------------------------*)
PROCEDURE SoundKanalON;
BEGIN
  INLINE($E4/$61/              (* IN   AL,61 *)
         $0C/$03/              (* OR   AL,03 *)
         $E6/$61)              (* OUT  61,AL *)
END;
(*-------------------------------------------------------------------------*)
PROCEDURE NoSound;
BEGIN
  INLINE($E4/$61               (* IN  AL,61 *)
        /$24/$FC               (* AND AL,FC *)
        /$E6/$61)              (* OUT 61,AL *)
END;
(*-------------------------------------------------------------------------*)
PROCEDURE Sound(Freq:INTEGER);
VAR CounterSet:INTEGER;
      MSB,LSB:BYTE;
BEGIN
  SoundKanalON;
  IF Freq < 37 THEN Freq := 37;
  CounterSet := ROUND(1.19318e6 / Freq);
  LSB := CounterSet AND $00FF;
  MSB := (CounterSet SHR 8) AND $00FF;
  Port[$43] := $B6;
  Port[$42] := LSB;
  Port[$42] := MSB
END;
(*-------------------------------------------------------------------------*)
PROCEDURE Beep(Anzahl:INTEGER);
VAR i : INTEGER;
BEGIN
   FOR i := 1 TO Anzahl DO Write(#7);
END;
```

Die Routinen wurden auf verschiedenen Systemen unter Turbo-Pascal 3.0 getestet

```
(*-------------------------------------------------------------------------*)
(* ROUTINEN zur Steuerung der HERCULES Karte                               *)
(*-------------------------------------------------------------------------*)
PROCEDURE HerculesMode(Option:BOOLEAN);
CONST
     IndexReg = $03B4;     (* i    6845 Index Register    *)
     DataReg  = $03B5;     (* i/o  6845 Data  Register    *)
     ModeCont = $03B8;     (* o    Display Mode Ctrl Port *)
     ConfigSw = $03BF;     (* o    Configuration Switch   *)
     TxtVal:ARRAY [0..11] OF BYTE =
         ($61,$50,$52,$0f,$19,$06,$19,$19,$02,$0d,$0b,$0c);
     GrpVal:ARRAY [0..11] OF BYTE =
         ($35,$2d,$2e,$07,$5b,$02,$57,$57,$02,$03,$00,$00);
VAR
  on,Blank,grph,txt,Attri,RegSel:INTEGER;
BEGIN
 IF Option THEN BEGIN
      grph  := 2 + 128;                  (* PAGE 1 *)
      on    := grph + 8;
      RegSel:= 0;
      PORT[ModeCont] := grph;            (* Mode Control Port *)
      REPEAT
         PORT[IndexReg] := RegSel;
         RegSel := SUCC(RegSel);
      UNTIL RegSel = 12;
      PORT[ConfigSw] := 3;               (* Full MODE   *)
      PORT[ModeCont] := on;              (* Einschalten *)
    END

 ELSE BEGIN
      Blank := $20;
      Attri := $0f;                      (* BLANK Value *)
      RegSel:= 0;
      txt   := $20;
      on    := txt + 8;
      PORT[ModeCont] := txt;             (* Mode Control Port *)
      REPEAT
        PORT[IndexReg] := RegSel;
        PORT[DataReg] := TxtVal[RegSel];
        RegSel := SUCC(RegSel);
      UNTIL RegSel = 12;
      PORT[ConfigSw] := 1;               (* Diag MODE   *)
      PORT[ModeCont] := on;              (* Einschalten *)
    END
END;
(*-------------------------------------------------------------------------*)
FUNCTION ByteOffset(X,Y:INTEGER):INTEGER;
BEGIN
   ByteOffset := $2000 * (Y MOD 4) + 90 * (Y DIV 4) + (X DIV 8)
END;
(*-------------------------------------------------------------------------*)
FUNCTION BitOffset(X:INTEGER):INTEGER;
BEGIN
   BitOffset := 8 - (X MOD 8)
END;
(*-------------------------------------------------------------------------*)
FUNCTION TestDot(X,Y:INTEGER):BOOLEAN;
CONST BitPos:ARRAY[1..8]OF BYTE = (1,2,4,8,16,32,64,128);
VAR Offs,Bitp,Value:INTEGER;
BEGIN
     Offs := ByteOffset(X,Y);
     Bitp := BitOffset(X);
     Value := MEM[$B800:Offs];
     IF  Value AND BitPos[BitP] = 0 THEN TestDot := False
                                    ELSE TestDot := True
END;
(*-------------------------------------------------------------------------*)
PROCEDURE SetResDot(x,y:INTEGER;SetRes:BOOLEAN);
CONST BitPos:ARRAY[1..8]OF BYTE = (1,2,4,8,16,32,64,128);
VAR   Offs,Bitp,Value:INTEGER;
BEGIN
     Offs := ByteOffset(X,Y);
     Bitp := BitOffset(X);
     Value := MEM[$B800:Offs];
     IF SetRes = True THEN MEM[$B800:Offs] := (Value OR BitPos[BitP])
        ELSE MEM[$B800:Offs] := (Value AND (255 - BitPos[BitP]))
END;
(*-------------------------------------------------------------------------*)
PROCEDURE SetResByte(x,y:INTEGER;SetRes:BOOLEAN);
VAR   Offs,Value:INTEGER;
BEGIN
     Offs := ByteOffset(X,Y);
     IF SetRes = True THEN MEM[$B800:Offs] := $FF
                      ELSE MEM[$B800:Offs] := $00
END;
(*-------------------------------------------------------------------------*)
PROCEDURE FillSegment(X,Y,AnzX,AnzY:INTEGER;SetRes:BOOLEAN);
VAR Offs,Xpos,Ypos:INTEGER;
    Value :BYTE;
BEGIN
     AnzX := AnzX DIV 8;
     IF SetRes  THEN Value := $FF
                ELSE Value := $00;
     Xpos := 0;
     Ypos := 0;
     REPEAT
        Offs := ByteOffset(X+Xpos,Y+Ypos);
          REPEAT
             MEM[$B800:Offs+Xpos] := Value;
             Xpos := Succ(Xpos);
          UNTIL AnzX < Xpos;
        Xpos := 0;
        Ypos := Succ(Ypos);
     UNTIL AnzY < Ypos
END;
(*-------------------------------------------------------------------------*)
PROCEDURE SwitchDot(x,y:INTEGER);
CONST BitPos:ARRAY[1..8]OF BYTE = (1,2,4,8,16,32,64,128);
VAR   Offs,Bitp,Value:INTEGER;
BEGIN
     Offs := ByteOffset(X,Y);
     Bitp := BitOffset(X);
     Value := MEM[$B800:Offs];
     MEM[$B800:Offs] := (Value XOR BitPos[BitP])
END;
(*-------------------------------------------------------------------------*)
PROCEDURE ClearHercules;
BEGIN
   INLINE($06/                 (* PUSH ES *)
          $57/                 (* PUSH DI *)
          $B8/$00/$B8/         (* MOV AX,B800  ;Startadr. Grafik RAM *)
          $8E/$C0/             (* MOV ES,AX *)
          $31/$FF/             (* XOR DI,DI *)
          $B9/$00/$40/         (* MOV CX,4000 *)
          $31/$C0/             (* XOR AX,AX *)
          $F3/$AB/             (* REPZ  STOSW *)
          $5F/                 (* POP  DI *)
          $07)                 (* POP  ES *)
END;
(*-------------------------------------------------------------------------*)
PROCEDURE HerculesWrite(X,Y,Attr:INTEGER;ActChar:CHAR);
TYPE ZchnArr = ARRAY [0..7,0..7] OF BYTE;
CONST ZchnGen :ZchnArr = (($00,$00,$CC,$00,$CC,$CC,$78,$00),  (* ü 129*)
                          ($CC,$00,$FC,$0C,$FC,$CC,$FC,$00),  (* ä 132*)
                          ($00,$CC,$00,$78,$CC,$CC,$78,$00),  (* ö 148*)
                          ($CC,$30,$78,$CC,$FC,$CC,$CC,$00),  (* Ä 142*)
```

▼

```
                        ($CC,$30,$48,$CC,$CC,$48,$30,$00),   (* Ö 153*)
                        ($CC,$00,$CC,$CC,$CC,$CC,$78,$00),   (* Ü 154*)
                        ($38,$64,$64,$78,$64,$64,$78,$60),   (* ß 225*)
                        ($AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA));
VAR I,Offs,CharOffs,ArrPos:INTEGER;
    TxtTblOrg:ARRAY[0..127,0..7] OF BYTE ABSOLUTE $FFA6:$E;
BEGIN
 CharOffs := Ord(ActChar);
 IF CharOffs <= 127 THEN BEGIN
  CASE Attr OF
    1 : BEGIN
          FOR I := 0 TO 7 DO BEGIN
            Offs := ByteOffset(X,Y+i);
            MEM[$B800:Offs]:=TxtTblOrg[CharOffs,i]
          END
        END;
    2 : BEGIN
          FOR I := 0 TO 7 DO BEGIN
            Offs := ByteOffset(X,Y+i);
            MEM[$B800:Offs]:=NOT TxtTblOrg[CharOffs,i]
          END
        END;
    3 : BEGIN
          FOR I := 0 TO 6 DO BEGIN
            Offs := ByteOffset(X,Y+i);
            MEM[$B800:Offs]:=TxtTblOrg[CharOffs,i]
          END;
          Offs := ByteOffset(X,Y+7);
          MEM[$B800:Offs]:=$FF
        END
 END (*Case*)
END  (*IF BEGIN*)

ELSE BEGIN
      CASE Ord(ActChar) OF
        129 : ArrPos := 0;
        132 : ArrPos := 1;
        148 : ArrPos := 2;
        142 : ArrPos := 3;
        153 : ArrPos := 4;
        154 : ArrPos := 5;
        225 : ArrPos := 6;
        ELSE  ArrPos := 7;
      END;
      CASE Attr OF
        1 : BEGIN
             FOR I := 0 TO 7 DO BEGIN
                Offs := ByteOffset(X,Y+i);
                MEM[$B800:Offs]:=ZchnGen[ArrPos,I]
             END
            END;
        2 : BEGIN
            FOR I := 0 TO 7 DO BEGIN
                Offs := ByteOffset(X,Y+i);
                MEM[$B800:Offs]:=NOT ZchnGen[ArrPos,I]
             END
            END;
        3 : BEGIN
            FOR I := 0 TO 6 DO BEGIN
                Offs := ByteOffset(X,Y+i);
                MEM[$B800:Offs]:= ZchnGen[ArrPos,I]
            END;
            Offs := ByteOffset(X,Y+7);
            MEM[$B800:Offs]:=$FF
     END
 END (*ELSE BEGIN*)
END (*CASE*)
END;
```

```
(*-------------------------------------------------------------------------*)
PROCEDURE HerculesHardCopy(Invers:BOOLEAN);
VAR Offs,X,Y:INTEGER;
    PrtVal :BYTE;
BEGIN
   Write(lst,#27#65#7); (* 22/216 LineSpacing *)
   FOR X := 0 TO 89 DO BEGIN
       Write(lst,#27#75#92#1);  (* Bit Image Mode 348 Images*)
       FOR Y := 347 DOWNTO 0 DO BEGIN
          Offs := ByteOffset(X*8,Y);
          PrtVal := MEM[$B800:Offs];
          IF Invers THEN PrtVal := NOT PrtVal;
          Write(lst,chr(PrtVal))
       END;
       Write(lst,#13#10) (* Return LineFeed *)
   END;
   Write(lst,#13#10#12#27#50) (* FormFeed 1/6 LineSpacing*)
END;
(*-------------------------------------------------------------------------*)
PROCEDURE CopyPage;
BEGIN
  INLINE
       ($1E/                  (* PUSH    DS          *)
        $56/                  (* PUSH    SI          *)
        $06/                  (* PUSH    ES          *)
        $57/                  (* PUSH    DI          *)
        $FC/                  (* CLD                 *)
        $B9/$00/$B0/          (* MOV     CX,B000   Quelle DS:SI*)
        $8E/$D9/              (* MOV     DS,CX       *)
        $31/$F6/              (* XOR     SI,SI       *)
        $B9/$00/$B8/          (* MOV     CX,B800   Ziel ES:DI *)
        $8E/$C1/              (* MOV     ES,CX       *)
        $31/$FF/              (* XOR     DI,DI       *)
        $B9/$00/$40/          (* MOV     CX,4000   32 KByte *)
        $F3/                  (* REPZ                *)
        $A5/                  (* MOVSW               *)
        $5F/                  (* POP     DI          *)
        $07/                  (* POP     ES          *)
        $5E/                  (* POP     SI          *)
        $1F);                 (* POP     DS          *)
END;
(*-------------------------------------------------------------------------*)
PROCEDURE PutGetPic(PicFile:CHAR;Save:BOOLEAN);
VAR i,Offs:INTEGER;
    VideoFile :FILE;
    VideoPuffer:ARRAY[0..$7FFF] OF BYTE;
BEGIN
   IF Save THEN BEGIN
      ASSIGN(VideoFile,PicFile+'.PIC');
      REWRITE(VideoFile); (* Erstellen des Files *)
      FOR Offs := 0 TO $7FFF DO
          VideoPuffer[Offs]:=MEM[$B800:Offs];
      BlockWrite(VideoFile,VideoPuffer,256);
      CLOSE(VideoFile)
     END
   ELSE BEGIN
      ASSIGN(VideoFile,PicFile+'.PIC');
      (*$I-*)RESET(VideoFile);(*$I+*)
      IF IOresult = 0 THEN
         BlockRead(VideoFile,VideoPuffer,256);
      CLOSE(VideoFile);
      FOR Offs :=0 TO $7FFF DO   (* Bild nach Page 1 laden *)
          MEM[$B000:Offs]:=VideoPuffer[Offs];
      CopyPage;                  (* Bild nach Page 2 kopieren *)
      END
END;
(*************************************************************************)
```

HerculesWrite(x,y,Attribut:Integer; ActChar:Char)
Die Ausgabe von Text im Hires-Modus wird durch diese Prozedur möglich. Die Routine verwendet den Zeichengenerator des Betriebssystems ab Adresse FFA6h:E000h. Leider sind in diesem Bereich nur die Muster der Zeichen mit den Codes 0 bis 127 abgelegt. Für die deutschen Sonderzeichen wurde deshalb eine kleine Erweiterung vorgenommen. Der Umfang des Arrays ist jederzeit erweiterbar. Sollten auf Ihrem Schirm seltsame Zeichen erscheinen, so muß nur die erwähnte Adresse geändert werden.

Attribut 1: dunkler Untergrund.
2: heller Untergrund.
3: dunkler Untergrund unterstrichen.

HerculesHardCopy(Invers:Boolean)
Die Hardcopy des Hires-Bereiches ist für einen Epson-FX-80 oder einen CP80 vorgesehen. Das Bild wird um 90 Grad gedreht ausgegeben.

CopyPage
Routine für schnelles Kopieren der unteren Video-Page auf die obere Seite.

PutGetPic(PicFile:Char;Save:Boolean)
Das Speichern und Laden vollständiger Bildschirminhalte ist mit dieser Prozedur möglich. Als Dateiname wird ein Buchstabe verwendet, dem intern die Extension .PIC zugeteilt wird. Das Bild wird zuerst in die nicht sichtbare Seite geladen und dann schnell auf die sichtbare übertragen.
Achtung: Diese Routine darf nur verwendet werden, solange die Hires-Grafik aktiv ist, da die Lowres-Page als Zwischenspeicher dient. Mit dieser Prozedur wird eine „Dia-Show" möglich. Das Bild wird zuerst mit der langsamen Laderoutine in die untere Seite geladen und dann schnell auf die sichtbare Seite kopiert.
Vor Verlassen des Grafik-Modus muß mit ClrScr der Lowres-Bereich initialisiert werden. Es besteht natürlich auch die Möglichkeit, die 4 KByte zwischenzuspeichern und so den alten Bildschirminhalt zu retten.

Literatur

[1] Technisches Handbuch, IBM.
[2] Monochrome Graphic Printer Version II Manual, IBM.
[3] Turbo-Pascal-Handbuch, Heimsoeth.

Taschenrechner mit DEBUG eingeben

In der Dezember-Ausgabe wurde in dem Beitrag „Taschenrechner auf Knopfdruck" eine falsche Methode der Programmeingabe mit DEBUG geschildert. Das abgedruckte Assemblerlisting kann nur von Lesern verwendet werden, die einen Assembler besitzen, denn alle mit R gekennzeichneten Stellen werden vom Linker nochmals überarbeitet. Der Linker erzeugt eine lauffähige EXE-Datei, die mit EXE2BIN in eine COM-Datei umgewandelt werden kann. Unser Bild zeigt den Hex-Dump der COM-Datei. Die Eingaben lauten:

A>DEBUG	-N CALC.COM
-E 100 E9 42 04 usw.	-W
-R CX	-Q
:47E	

D. Strauß

```
242B:0100  E9 42 04 0A E4 74 05 2E-FF 2E CB 04 9C 2E FF 1E   iB..dt....K.....
242B:0110  CB 04 3D 00 78 74 01 CF-2E 3A 06 D5 04 75 F8 1E   K.=.xt.O.:.U.ux.
242B:0120  06 8C C8 8E D8 8E C0 8C-16 C7 04 89 26 C9 04 8E   ..H.X.@..G..&I..
242B:0130  D0 BC FE 00 FB 55 53 51-52 56 57 FE 0E D5 04 E8   P<~.{USQRVW~.U.h
242B:0140  E8 01 E8 8E 01 F8 9C E8-0A 02 BE 11 04 E8 34 01   h.h..x.h..>...h4.
242B:0150  9D 73 03 BE 32 04 E8 2B-01 B8 0A 00 87 06 D3 04   .s.>2.h+.8....S.
242B:0160  E8 5E 02 87 06 D3 04 B0-0D E8 22 01 A1 D1 04 E8   h^...S.O.h".!Q.h
242B:0170  4F 02 B0 0D E8 17 01 33-F6 32 E4 CD 16 FB 3D 00   O.O.h..3v2dM.{=.
242B:0180  4B 75 02 B0 08 3C 61 72-02 2C 20 3C 1B 74 67 3C   Ku.O.<ar.,.<.tg<
242B:0190  51 74 63 3C 08 75 0C 0B-F6 74 DE 4E B0 08 E8 ED   Qtc<.u..vt^NO.hm
242B:01A0  00 EB D6 3C 30 72 36 8A-26 D3 04 80 FC 02 75 04   .kV<0r6.&S..|.u.
242B:01B0  3C 31 77 29 80 FC 08 75-04 3C 37 77 20 3C 39 76   <1w).|.u.<7w <9v
242B:01C0  0D 80 FC 0A 74 17 3C 41-72 13 3C 46 77 0F 83 FE   ..|.t.<Ar.<Fw..~
242B:01D0  10 73 A6 88 84 D6 04 E8-B4 00 46 EB 9C 3C 3F 74   .s&..V.h4.Fk.<?t
242B:01E0  29 3C C1 74 25 33 FF 3A-85 E4 03 74 0B 47 81 FF   )<At%3.:.d.t.G..
242B:01F0  0F 00 72 F3 EB 83 EB 2E-E8 A0 01 D1 E7 A1 D1 04   ..rsk.k.h .Qg!Q.
242B:0200  FF 95 F3 03 A3 D1 04 E9-3C FF E8 47 01 BE 41 04   ..s.#Q.i<.hG.>A.
242B:0210  E8 71 00 32 E4 CD 16 FB-E8 39 01 E8 66 00 32 E4   hq.2dM.{h9.hf.2d
242B:0220  CD 16 FB E9 20 FF E8 42-01 C6 06 D5 04 00 5F 5E   M.{i .hB.F.U.._^
242B:0230  5A 59 5B 5D 32 E4 8E 16-C7 04 8B 26 C9 04 07 1F   ZY[]2d..G..&I...
242B:0240  E9 C9 FE 8B C1 C3 23 C1-C3 0B C1 C3 33 C1 C3 35   iI~.AC#AC.AC3AC5
242B:0250  FF FF 40 C3 D1 E8 C3 D1-E0 C3 03 C1 C3 2B C1 C3   ..@CQhCQ`C.AC+AC
242B:0260  F7 E1 C3 33 D2 0B C9 74-08 0B C0 74 02 F7 F1 F8   waC3R.It..@t.wqx
242B:0270  C3 F9 C3 D1 EF 8A 95 E0-03 88 16 D3 04 0B C9 74   CyCQo..`...S..It
242B:0280  02 8B C1 C3 AC 3C 00 74-FA E8 02 00 EB F6 3C 08   ..AC,<.tzh..kv<.
242B:0290  75 0A E8 07 00 B0 20 E8-02 00 B0 08 51 50 B4 03   u.h..0 h..0.QP4.
242B:02A0  B7 00 CD 10 58 3C 0D 74-1C 3C 08 74 1E 50 80 FA   7.M.X<.t.<.t.P.z
242B:02B0  42 72 04 B2 31 FE C6 FE-C2 58 B9 01 00 B3 70 B4   Br.21~F~BX9..3p4
242B:02C0  09 CD 10 EB 08 FE C6 B2-32 EB 02 FE CA B4 02 CD   .M.k.~F22k.~J4.M
242B:02D0  10 59 C3 BA 31 06 B0 C9-E8 3C 00 FE C2 B9 11 00   .YC:1.0Ih<.~B9..
242B:02E0  B0 CD E8 35 00 B6 0B E8-30 00 BA 31 07 B0 BA E8   0Mh5.6.h0.:1.0:h
242B:02F0  25 00 FE C6 80 FE 0A 76-F6 B2 43 FE CE E8 17 00   %.~F.~.vv2C~Nh..
242B:0300  80 FE 07 77 F6 FE CE B0-BB E8 0B 00 B6 0B B0 BC   .~.wv~N0;h..6.0<
242B:0310  E8 04 00 B2 31 B0 C8 B9-01 00 50 51 B4 02 CD 10   h..210H9..PQ4.M.
242B:0320  59 58 BB 70 00 B4 09 CD-10 C3 B4 03 32 FF CD 10   YX;p.4.M.C4.2.M.
242B:0330  89 16 CF 04 B6 06 BF E6-04 B2 31 52 B4 02 CD 10   ..O.6.?f.21R4.M.
242B:0340  B4 08 CD 10 5A AB FE C2-80 FA 44 72 EE FE C6 80   4.M.Z+~B.zDrn~F.
242B:0350  FE 0C 72 E5 B8 00 06 B9-32 07 BA 42 0A B7 70 CD   ~.re8..92.:B.7pM
242B:0360  10 BA 32 07 B7 00 B4 02-CD 10 C3 B6 06 BE E6 04   .:2.7.4.M.C6.>f.
242B:0370  B7 00 B2 31 B4 02 52 CD-10 AD 8A DC B9 01 00 B4   7.214.RM.-.\9..4
242B:0380  09 CD 10 5A FE C2 80 FA-44 72 E9 FE C6 80 FE 0C   .M.Z~B.zDri~F.~.
242B:0390  72 E0 8B 16 CF 04 B4 02-CD 10 C3 BD 01 00 33 C9   r`..O.4.M.C=..3I
242B:03A0  4E 78 1D 8A 84 D6 04 2C-30 3C 09 76 02 2C 07 32   Nx...V.,0<.v.,.2
242B:03B0  E4 F7 E5 03 C8 8B C5 F7-26 D3 04 8B E8 4E 79 E3   dwe.H.Ew&S..hNyc
242B:03C0  C3 33 C9 50 33 D2 F7 36-D3 04 52 41 0B C0 75 F4   C3IP3Rw6S.RA.@ut
242B:03D0  58 04 30 3C 39 76 02 04-07 E8 B2 FE E2 F2 58 C3   X.0<9v...h2~brXC
242B:03E0  02 08 0A 10 49 4A 5A 48-55 4F 4B 58 52 4C 2B 2D   ....IJZHUOKXRL+-
242B:03F0  2A 2F 0D 73 02 73 02 73-02 73 02 46 02 49 02 4F   */.s.s.s.s.F.I.O
242B:0400  02 4C 02 54 02 57 02 5A-02 5D 02 60 02 63 02 43   .L.T.W.Z.].`.c.C
242B:0410  02 51 3D 45 4E 44 45 20-20 55 50 4E 2D 43 41 4C   .Q=ENDE  UPN-CAL
242B:0420  43 20 00 3F 3D 48 69 6C-66 65 20 42 61 73 69 73   C .?=Hilfe Basis
242B:0430  3D 00 9A 42 45 52 4C 2E-21 20 42 61 73 69 73 3D   =..BERL.! Basis=
242B:0440  00 42 41 53 49 53 3A 20-20 5A 3D 31 30 20 48 3D   .BASIS:  Z=10 H=
242B:0450  31 36 20 20 20 20 20 20-20 20 49 3D 32 20 20 4A   16         I=2  J
242B:0460  3D 38 20 41 72 69 74 68-6D 2E 3A 2B 20 2D 20 2A   =8 Arithm.:+ - *
242B:0470  20 2F 0D 55 3D 55 6E 64-20 20 4F 3D 4F 64 65 72    /.U=Und  O=Oder
242B:0480  20 20 1A 00 58 3D 45 78-6B 6C 75 73 69 76 2D 4F     ..X=Exklusiv-O
242B:0490  64 65 72 0D 4B 3D 32 65-72 2D 4B 6F 6D 70 6C 65   der.K=2er-Komple
242B:04A0  6D 65 6E 74 20 4C 2C 52-3D 53 68 69 66 74 20 6C   ment L,R=Shift l
242B:04B0  69 2E 2F 72 65 2E 43 52-3D 45 6E 74 65 72 20 20   i./re.CR=Enter
242B:04C0  51 3D 45 6E 64 65 00 00-00 00 00 00 00 00 00 00   Q=Ende..........
242B:04D0  00 00 00 0A 00 00 00 00-00 00 00 00 00 00 00 00   ................
242B:04E0  00 00 00 00 00 00 0D 0A-55 50 4E 2D 43 41 4C 43   ........UPN-CALC
242B:04F0  20 31 2E 30 20 28 43 29-20 53 48 41 4D 52 4F 43    1.0 (C) SHAMROC
242B:0500  4B 20 53 4F 46 54 57 41-52 45 20 20 20 20 20 41   K SOFTWARE     A
242B:0510  75 66 72 75 66 3A 20 41-4C 54 2D 31 0D 0A 24 0D   ufruf: ALT-1..$.
242B:0520  0A 55 50 4E 2D 43 41 4C-43 20 69 73 74 20 62 65   .UPN-CALC ist be
242B:0530  72 65 69 74 73 20 69 6E-73 74 61 6C 6C 69 65 72   reits installier
242B:0540  74 21 0D 0A 24 B8 16 35-CD 21 89 1E CB 04 8C 06   t!..$8.5M!..K...
242B:0550  CD 04 BE 03 01 8B FB B9-64 00 F3 A6 74 14 BA 03   M.>...{9d.s&t...
242B:0560  01 B8 16 25 CD 21 BA E6-04 B4 09 CD 21 BA CA 05   .8.%M!:f.4.M!:J.
242B:0570  CD 27 BA 1F 05 B4 09 CD-21 B8 00 4C CD 21         M':..4.M!8.LM!H.
```

Mit DEBUG läßt sich der Hex-Dump der COM-Datei CALC.COM eingeben

Dipl.-Ing. Manfred Keul

DOS-EXEC-Funktion von Turbo-Pascal aus

Wie man Fremdprogramme aufruft und wieder nach Turbo-Pascal zurückkehrt

Turbo-Pascal bietet mehrere Möglichkeiten, von einem Hauptprogramm aus andere Programmteile von einer Festplatte in den Arbeitsspeicher zu laden und abzuarbeiten. Leider funktionieren alle diese Verfahren nur für spezielle, von Turbo-Pascal erzeugte Dateien. Aber mit einem Trick geht es auch mit x-beliebigen DOS-Programmen.

Der einfachste Weg, Programme von Turbo-Pascal aus aufzurufen, führt über die Verwendung von Overlays: Mehrere Module eines Programms teilen sich einen gemeinsamen Speicherbereich in der Form, daß zu jedem Zeitpunkt immer nur eines dieser (Overlay-)Module im Speicher steht. Befindet sich beispielsweise zur Zeit ein Modul A im gemeinsamen Speicher und wird während des Programmablaufs das Modul B benötigt, so wird dieses von der Festplatte geladen und überlagert das Modul A.

Die zweite und dritte Möglichkeit: Execute und Chain. Beide Prozeduren erlauben es, ein anderes Turbo-Pascal-Programm in den Speicher zu laden und dieses dann auszuführen. Der Unterschied liegt darin, daß Execute für normale Turbo-Programme (COM-Files) eingesetzt wird, während Chain spezielle CHN-Files voraussetzt. Das sind Turbo-Programme, die keine eigene Runtime-Library besitzen, sondern auf die schon im Speicher befindliche Library des sie ladenden Programms zugreifen.

Keine der drei angeführten Alternativen (auch nicht Execute!) erlaubt es aber, ein x-beliebiges Programm, das nicht vom Turbo-Compiler erzeugt wurde, auszuführen. Außerdem bieten Chain und Execute keine Möglichkeit, wieder in das aufrufende Programm zurückzukehren. DOS stellt für diese Zwecke zwar eine entsprechende Funktion zur Verfügung (Nummer 4BH, EXEC), aber sogar die Techniker des Turbo-Mutterhauses Borland in Amerika behaupten, daß sich diese Funktion nicht von Turbo-Pascal aus nutzen läßt.

Im folgenden zeigen wir, daß es doch geht – wenngleich die Methode ein wenig mühsam erscheinen mag.

Das Beispielprogramm (*Bild*) ruft von Turbo-Pascal aus den Debugger Debug auf, mit Autoexec.bat als Befehlszeilen-

```
program EXECDEMO;  ( demonstrates EXEC call from within TURBO program )

const STACKSIZE = $100;     ( stack our program needs (in paragraphs); must be
                    entered as "Minimum" and "Maximum Stack Size" before
                    generating .COM-file. )

type
     REGISTER = record
              AX,BX,CX,DX,BP,SI,DI,DS,ES,FLAGS : integer;
     end;

     NAMESTRING = string[80];


var REG               : REGISTER;

{*************************************************************************}

procedure SETBLOCK;

(   release all the memory our program doesn't need )

begin
with REG do
     begin
     BX:= Sseg - Cseg + STACKSIZE;      ( BX points to our end of stack,
                                          release everything above )
     ES := CSEG ;                       ( segment of our memory block )
     AX:= $4A00;                        ( SETBLOCK )
     MSDOS(REG);

     if odd(FLAGS) then
        begin
        writeln ('SETBLOCK error!, AX = ',AX);
        halt;
        end;
     end;
end;    ( SETBLOCK )

{*************************************************************************}

procedure EXEC (PROGRAMNAME, COMMANDLINE: NAMESTRING);

( gives control to program PROGRAMNAME with COMMANDLINE parameters,
uses DOS function 4BH: EXEC.  )
```

Dieses Programm demonstriert den EXEC-Aufruf von Turbo-Pascal aus

```
type  PAR_TYPE = record
                 ENV_SEG,COMM_OFF,COMM_SEG,
                 D1,D2,D3,D4                : integer;
       end;

const                ( EXEC parameter block )
    PAR_BLOCK    : PAR_TYPE = (ENV_SEG :0; COMM_OFF :0;COMM_SEG :0;
                               D1:0;D2:0;D3:0;D4:0);

( defining the following variables as "typed constants" forces them to
reside in the code segment, this simplifies assembler access to them )

    ASCIIZ : NAMESTRING ='';
    SSSAVE  : integer = 0;
    SPSAVE  : integer = 0;
    AXREG   : integer = 0;            (######)
    FLAGSREG: integer = 0;            (######)

begin
ASCIIZ := PROGRAMNAME + chr(0);     ( make ASCIIZ string )
COMMANDLINE := COMMANDLINE + chr(13);

with PAR_BLOCK do              ( fill in commandline into parameter block )
     begin
     COMM_SEG := seg (COMMANDLINE);
     COMM_OFF := ofs (COMMANDLINE);
     end;

{-------------------------------------------------------------------------}

( Calling EXEC through TURBO's MsDos procedure causes trouble with DOS 2.1,
so we code this by hand: )

inline

  ( first, save registers DS,BP,SP,SS: )
  ($1E/                       (  PUSH DS        )
  $55/                        (  PUSH BP        )
  $2E/                        (  CS:            )
  $89/$26/SPSAVE/             (  MOV [SPSAVE],SP )
  $8C/$D0/                    (  MOV AX,SS      )
  $2E/                        (  CS:            )
  $A3/SSSAVE/                 (  MOV [SSSAVE],AX )

  ( copy CS into DS and ES: )
  $0E/                        (  PUSH CS        )
  $0E/                        (  PUSH CS        )
  $1F/                        (  POP DS         )
  $07/                        (  POP ES         )

  ( for EXEC, DS:DX must point to ASCIIZ string with filename to be executed,
  so we set DX to 2nd Byte of ASCIIZ (1st byte contains string length): )
  $BA/ASCIIZ+1/               (  MOV DX,ASCIIZ + 1    )

  ( ES:BX must point to parameter block containing command line etc. )
  $BB/PAR_BLOCK/              (  MOV BX,PAR_BLOCK     )

  $B8/$00/$4B/                (  MOV AX,4B00, EXEC function code    )
  $CD/$21/                    (  INT 21, do it )

  ( after child program has been executed, we first of all have to restore
  our registers: )
  $2E/                        (  CS:            )
  $8B/$1E/SSSAVE/             (  MOV BX,[SSSAVE] )
  $8E/$D3/                    (  MOV SS,BX      )
  $2E/                        (  CS:            )
  $8B/$26/SPSAVE/             (  MOV SP,[SPSAVE] )
  $5D/                        (  POP BP         )
  $1F/                        (  POP DS         )

  ( AX and Flagregister contain EXEC error codes, so we save them: )
  $2E/                        (  CS:            )
  $A3/AXREG/                  (  MOV [AXREG],AX )
  $9C/$58/                    (  PUSHF, POP AX  )
  $2E/                        (  CS:            )
  $A3/FLAGSREG);              (  MOV [FLAGSREG],AX )

{-------------------------------------------------------------------------}

( for DOS 3.0, code between the dashed lines may be replaced by

with REG do
     begin
     DS := seg (ASCIIZ);
     DX := ofs (ASCIIZ) + 1;
     ES := seg (PAR_BLOCK);
     BX := ofs (PAR_BLOCK);
     AX := $4B00;
     MsDos;
     end;

In this case, the three "######" marked lines must be omitted.
However, this DOS 3.0 feature is not documented! )

with REG do
     begin
     AX := AXREG;  FLAGS := FLAGSREG;        (######)
     if odd (FLAGS) then
        begin
        writeln ('EXEC error!, AX = ',AX);
        halt;
        end;

     end;    ( with )
end;  ( EXEC )


{*************************************************************************}
{*****                                                               *****}
{*****                    M A I N   P R O G R A M                    *****}
{*****                                                               *****}
{*************************************************************************}

begin
SETBLOCK;                                  ( deallocate memory )
writeln ('Calling debugger...');
EXEC ('\DOS\DEBUG.COM','\AUTOEXEC.BAT');    ( execute DEBUG \AUTOEXEC.BAT,
                                              DEBUG resides in directory \DOS )
writeln ('Debugger returned to TURBO program.');
writeln ('End of test.');
end.
```

Parameter. Genau wie nach Eingabe der Zeile
DEBUG AUTOEXEC.BAT ⟨return⟩
auf der DOS-Kommandoebene wird der Debugger mit der Aufgabe aufgerufen, Autoexec.bat zu untersuchen. (Sicherlich keine besonders intelligente Anwendung, aber es geht ja nur darum, das Prinzip zu zeigen.) Nachdem Debug vom Bediener beendet wurde (q-Kommando), geht die Kontrolle wieder zurück an das übergeordnete Turbo-Pascal-Programm.

Vorgehensweise

Die erste Schwierigkeit, die es zu überwinden gilt, ist die, daß ein Turbo-Pascal-Programm, wie jede andere COM-Datei auch, beim Start vom Betriebssystem den gesamten im Computer verfügbaren Speicherbereich zugewiesen bekommt.

Und Turbo-Pascal-Programme geben von sich aus nichts von diesem, normalerweise völlig überdimensionierten, Speicherblock zurück. Statt dessen legen sie im unteren Adreßbereich ihren Programmcode und ihre globalen Variablen ab, auf den obersten Adressen (maximal 64 KByte) richten sie sich einen Stack ein, und alles, was dazwischen liegt, wird dem Heap zugewiesen. So kommt es, daß überhaupt kein Speicherplatz mehr zur Verfügung steht, in dem man noch ein anderes Programm ausführen könnte. Im Turbo-Options-Menü gibt es allerdings das „I"- und „A"-Kommando, die es erlauben, die maximale Größe von Stack und Heap zu beschränken. (Die „I"- und „A"-Option erscheinen, nachdem man im Options-Menü „C" für COM-Datei eingegeben hat.

Stack und Heap werden in diesem Menü unter der Bezeichnung „dynamic memory" zusammengefaßt.)

Daher ist das erste, was wir zu tun haben – und hier liegt das einzige Problem dieses Verfahrens –, abzuschätzen, wieviel Stack- und Heap-Platz unser Programm maximal braucht. Dabei ist zu beachten, daß Turbo-Pascal die Parameter über den Stack übergibt, d. h. wir brauchen mindestens soviel „dynamic memory", um den umfangreichsten Parametersatz, den wir irgendeiner procedure oder function unseres Programms übergeben, unterzubringen. Haben wir diesen Wert festgelegt, so wandeln wir ihn in eine Hex-Zahl um. Da Turbo-Pascal die Angabe von Speichergrößen in Paragraphen, also 16-Byte-Blöcken, erwartet, müssen wir die Hex-Zahl durch 16 teilen, was wir näherungsweise durch Wegstreichen der letzten Ziffer erreichen. Die so ermittelte Zahl geben wir, bevor wir unser Programm compilieren, sowohl als Minimum wie auch als Maximum Stack Size im Turbo-Options-Menü ein. Des weiteren weisen wir diese Zahl einer Konstanten in unserem Turbo-Pascal-Programm, z. B. STACKSIZE, zu. Im Beispielprogramm wurden 4096 Byte für das „dynamic memory" veranschlagt, das ist bei weitem ausreichend für Programme, die nicht rekursiv sind, die keinen Heap brauchen (weil sie keine Pointer einsetzen) und die keine riesigen Felder oder Strukturen als Parameter an procedures oder functions übergeben. Die Rechnung sieht dann so aus:

4096 dezimal = 1000 hexadezimal
= 100 Paragraphen.

Als zweiten Schritt müssen wir jetzt dafür sorgen, daß unser Turbo-Pascal-Programm den gesamten Speicher oberhalb seines Stacks wieder an DOS zurückgibt, damit dorthin ein anderes Programm geladen werden kann. Für diesen Zweck stellt uns DOS die Funktion 4AH (SETBLOCK) zur Verfügung. Ihr wird im BX-Register die Zahl der Paragraphen übergeben, die wir behalten möchten, die also nicht für andere Aufgaben zur Verfügung stehen. In unserem Fall heißt das, daß BX folgenden Wert erhalten muß: die Differenz zwischen dem letzten Segment, das unser Programm belegt, und seinem ersten Segment plus STACKSIZE – der Größe unseres Stacks.

Gemäß der Speicher-Aufteilung von Turbo-Pascal ist sein erstes Segment das Codesegment, sein letztes das Stacksegment, die effektiven Adressen dieser Segmente liefern uns die Turbo-Pascal-Funktionen Cseg und Sseg. Die vollständige Sequenz zur Freigabe des von uns nicht benötigten Speichers lautet demnach:

```
BX := Sseg – Cseg + STACKSIZE;

ES := Cseg;        ES muß auf unser Code-
                   segment zeigen

AX := $4A00;       SETBLOCK-Funktions-
                   nummer

MsDos(REG);        ruft DOS auf
```

Im Anschluß daran können wir jedes Programm, das in den gerade freigegebenen Speicherbereich hineinpaßt, mit der DOS-Funktion 4BH (EXEC) ausführen.

Zu unserem Leidwesen tritt jetzt aber doch noch ein Problem auf: Es kann zu Schwierigkeiten führen, wenn wir die EXEC-Funktion über die Turbo-MsDos-Prozedur aufrufen, analog dazu, wie wir es soeben mit SETBLOCK gemacht haben. Eine Analyse des vom Turbo-Pascal-Compiler generierten Codes ergibt, daß er den Aufruf von MsDos(REG) zu einer Sequenz der folgenden Form übersetzt:

```
call setze_register
;kopiere den Inhalt von REG in die 8088-
Register

int 21H
;rufe DOS (hier: führe EXEC aus)

call hole_register
;kopiere 8088-Register nach REG.
```

Unglücklicherweise sagt das „DOS Technical Reference Manual", daß nach einem EXEC der Inhalt *aller* Register des Prozessors verloren ist – einschließlich der Stack-Register SS und SP. Das bedeutet, daß zu dem Zeitpunkt, zu dem das Turbo-Pascal-System den obigen Hole-Register-Call ausführt und daher auf den Stack schreibt, dieser einen undefinierten Zustand hat – und in einer solchen Situation kann alles mögliche passieren, inklusive eines Systemabsturzes.

Es ist nun so, daß DOS-Version 2.1 die Register SS und SP tatsächlich zerstört, DOS 3.0 hingegen rettet, im Gegensatz zu den Angaben im „Reference Manual", diese beiden Register. Wenn wir uns auf dieses freundliche Entgegenkommen von DOS 3.0 verlassen, können wir die EXEC-Funktion über MsDos(REG) aufrufen, vorausgesetzt, unser Programm soll nur unter DOS 3.0 laufen. Da es aber nicht sehr beruhigend ist, sich auf undokumentierte Eigenschaften von DOS zu verlassen, und weil auch die Beschränkung auf die Version 3.0 nicht ganz einzusehen ist, wird in dem Beispielprogramm der EXEC-Aufruf von einigen Zeilen „Inline"-Maschinencode erledigt, die außer den Registern SS und SP auch noch die Register DS und BP retten, was für das ordnungsgemäße Arbeiten des Turbo-Pascal-Programms notwendig ist.

Literatur:

[1] Improving Turbo's I/O, Cole Brecheen, PC Tech Journal, 9/85.

Ulrich Cebulla

Grafik mit EGA

Teil 2

Genau genommen stellt das BIOS der EGA-Karte zur Programmierung des Grafik-Controllers nur zwei Befehle zur Verfügung, nämlich „setze Punkt“ (AH=0C) und „lese ob Punkt gesetzt“ (AH=0D). Die Karte selbst besitzt aber viele Register, die von einem Programm her erreichbar sind. Mit der richtigen Programmierung dieser Register kann man sehr bequem komplexe Grafik-Aktionen zusammenbauen. Allerdings können hier bei unterschiedlichen EGA-Karten Kompatibilitätsprobleme auftreten, denn diese „höheren Funktionen“ müssen unter Umständen unterschiedlich angesprochen werden.

Vielleicht ist es für Sie zunächst interesssant zu wissen, daß die Pascal-Prozedur Pset aus dem letzten Heft, Pset (x,y-,color), schneller ist als die BIOS-Funktion „setze Punkt“. *Bild 1* zeigt beide Routinen.

Adressierung der IBM-EGA-Register

Wie bereits im letzten Heft gezeigt, besitzt jedes IC der EGA-Karte je ein Adreß- (oder besser Indexregister) und ein Datenregister. In das Indexregister wird zunächst die Adresse (oder besser der Index) des zu adressierenden Datenregisters geladen. Damit ist das gewünschte Datenregister adressiert. Nun kann das Datenregister geladen werden (siehe Tabellen). Beispiel:

```
port($3CE):=5;
port ($3CF):=$FF;
```

Mit diesen zwei Instruktionen wird $FF in das Mode-Register des Grafik-Controllers geladen. Auf Ausnahmen von dieser Regel wird an entsprechender Stelle ausdrücklich hingewiesen (Attribute Controller). Ehe nun die Register des Grafik-Controllers beschrieben werden, muß an dieser Stelle erst ein Register des „Sequenzers“ erklärt werden, und zwar das „Map Mask Register“ (Index 02). Das Map Mask Register wird durch Laden des Indexes 02 in Adresse 3C4H unter Adresse 3C5H erreicht, wie es im Folgenden dargestellt ist:

Map Mask Register

Port.Index
3C4H.02
3C5H

Bit 0:
Erlaube schreiben in Speicherebene 0

Bit 1:
Erlaube schreiben in Speicherebene 1

Bit 2:
Erlaube schreiben in Speicherebene 2

Bit 3:
Erlaube schreiben in Speicherebene 3

Bit 4...7:
Nicht benutzt.

Dieses Register ist ein Filter, durch das alle Schreibbefehle zu den Speicherebenen laufen. Eine logische 1 in den Bits 0...3 erlaubt dem Prozessor das Schreiben in die entsprechende Speicherebene. Wird zum Beispiel in das Register der Wert 0FH geschrieben, so können alle 32 Bit eines Bildschirmspeicher-Wortes (8×4 Bits) in einem Speicherzyklus geschrieben werden. In dieses Register wird vom BIOS die Farbe gesetzt.

Die IBM-EGA-Grafik-Controller-Register

Alle Register des Grafik-Controllers sind sogenannte Write-Only-Register. Ihr Inhalt kann nur geändert, aber nicht gelesen werden.
Die Beschreibung enthält immer zunächst den Registernamen, dann die Portadresse für den Index (nur wenn nötig) und dann die Datenportadresse.

Grafik 1 Position

Port
3CCH

Dieses Register wird direkt adressiert. Nur Bit 0 und 1 sind benutzt. Muß immer 00H enthalten.

```
(* Filename: graphics.ega *)
const
     egabase=$A000;                    { Startadresse des Bildschirmspeichers }
var
   savemode,i,j :integer;

procedure Graphics(on:boolean);

type regs=record case integer of
             1:(ax,bx,cx,dx,bp,si,di,ds,es,flgs: integer);
             2:(al,ah,bl,bh,cl,ch,dl,dh: byte);
           end;
  var reg:regs;
begin
    if on                              { Grafiksmode einschalten }
    then begin
         reg.ax:=$0F00;                { AH = 0F lesen des EGA Status }
         intr($10,reg);
         savemode:=reg.al;             { EGA mode in AL }

         reg.ax:=$000F;                { EGA mode 0F Monochrome Bildschirm }
                                       { !!!!!    10 Enhanced Color Display }
         intr($10,reg);                { EGA BIOS Intrrupt $10 }
         end
    else begin                         { Grafiks Mode ausschalten }
         reg.ah:=0;
         reg.ax:=savemode;
         intr($10,reg);                { EGA BIOS Interrupt $10 }
         end;
end;
```

Bild 1. EGA auf zwei verschiedene Arten ansprechen

Grafik 2 Position

3CAH

Direkte Adressierung. Nur Bit 0 und 1 sind benutzt. Muß immer 01H enthalten.

Grafik 1,2 Adress Register

3CEH

Nur Bits 0...3 sind benutzt. Sie dienen als Index für die Grafik-Register 00 bis 08 des Grafik-Controllers.

Set/Reset Register

3CEH.00
3CFH

Nur Bits 0...3 werden benutzt.
Bit 0: Set/Reset Bit 0 (Speicherebene 0)
Bit 1: Set/Reset Bit 1 (Speicherebene 1)
Bit 2: Set/Reset Bit 2 (Speicherebene 2)
Bit 3: Set/Reset Bit 3 (Speicherebene 3)
Bit 4...7: Nicht benutzt

Diese Bits werden in die entsprechende Speicherebene geschrieben, wenn Write Mode 0 (siehe Mode Register) gesetzt ist. Im Enable Set/Reset Register kann zusätzlich noch ausgewählt werden, welche Speicherebenen wirklich beschrieben werden sollen. In dieses Register wird normalerweise die Farbe (Farbindex) abgelegt.

Enable Set/Reset Register

3CEH.01
3CFH

Bit 0: Erlaubt Set/Reset Speicherebene 0
Bit 1: Erlaubt Set/Reset Speicherebene 1
Bit 2: Erlaubt Set/Reset Speicherebene 2
Bit 3: Erlaubt Set/Reset Speicherebene 3
Bit 4...7: Nicht benutzt

Dieses Register ist ein sogenanntes Maskenregister für die Bits im Set/Reset Register. In Write Mode 0 kann über diese Bits ausgewählt werden, in welche Speicherebene(n) der Inhalt des Set/Reset Registers geschrieben wird. In Write Mode 0 wird in die Speicherebenen, die nicht erlaubt sind, der Wert geschrieben, der von der CPU kommt, jedoch jetzt abhängig vom Wert im Map Mask Register des Sequenzers.

Color Compare Register

3CEH.02
3CFH

Bit 0: Farbvergleich Speicherebene 0
Bit 1: Farbvergleich Speicherebene 1
Bit 2: Farbvergleich Speicherebene 2
Bit 3: Farbvergleich Speicherebene 3
Bit 4...7: Nicht benutzt

Diese 4 Bit repräsentieren eine Farbe, die mit dem Inhalt eines Bildschirmspeicher-Bytes verglichen werden kann.

Wird im Read Mode 1 (siehe Mode Register), ein Byte des Bildschirmspeichers gelesen, so erhält man in jeder Bitposition des Bytes eine 1, in der die Farbe des Bildschirmspeicher mit der im Color Compare Register übereinstimmt.

Data Rotate Register

3CEH.03
3CFH

Bit 0: Rotate Wert 0
Bit 1: Rotate Wert 1
Bit 2: Rotate Wert 2
Bit 3...4: Logische Funktionen
Bit 5...7: Nicht benutzt

Im Write Mode 0 werden die Bits des zu schreibenden Bytes um den binären Wert der Bits 0...2 nach rechts rotiert und erst dann in den Bildschirmspeicher geschrieben. Im Grafik-Controller befindet sich ein 32 Bit großes sogenanntes Prozessor-Latch-Register. Dieses Register wird bei jeder Leseoperation des Bildschirmspeichers neu geladen. Jede Schreiboperation zum Bildschirmspeicher kann mit den gerade gelesenen Daten wie folgt logisch verknüpft werden:

Bit
4 3
0 0 Keine Modifikation
0 1 UND-Funktion mit den vorher gelesenen Daten
1 0 ODER-Funktion mit den vorher gelesenen Daten
1 1 EXCLUSIV-ODER-Funktion mit den vorher gelesenen Daten.

Ist zur gleichen Zeit auch Rotation selektiert, so wird zuerst die Rotation und dann die Funktion ausgeführt.

Read Map Select Register

3CEH.04
3CFH

```
procedure pset(x,y,color:integer); { Setze einen Punkt an die Position x/y }
var offset    :integer;            { mit der Farbe = color                  }
    bitmask:byte;
begin

bitmask    := $80 shr (x and 7 );
offset     := (y * 80)+(x shr 3);
                              { GK = Grafiks-Kontroler }

     port[$3ce] := 0;         { GK SET/RESET Reg. 0 }
     port[$3cf] := color;     { Farbe ins SET/REST Register }
     port[$3ce] := 1;         { GK ENABLE SET/RESET Reg. 1 }
     port[$3cf] := $0F;       { erlaube alle Planes }

     port[$3ce] := 8;         { GK BIT MASK Reg. 8 }
     port[$3cf] := bitmask;   { PSET nur das durch x/y ausgewaehlte bit }

     mem[egabase:offset]:=mem[egabase:offset]; { read/write, siehe Text     }
                                               { ueber den Grafik Controller}

     port[$3cf]:=$FF;         { erlaube wieder alle bits in GK Reg. 8 }

     port[$3ce]:=1;           { GK ENABLE SET/RESET Reg. 8 }
     port[$3cf]:=0;           { disable alle Planes }
end;
{ BIOS, Punkt setzen }
procedure bset(x,y,color:integer); { Setze einen Punkt an die Stelle x/y      }
                                   { mit der Farbe = color (BIOS intr10 AH=0C) }
                                   { Diese Procedure ist 2,17 mal langsamer    }
                                   { als die Pascal "pset".                    }
                                   { Ist das bit 7 der color var 1, so wird    }
                                   { noch die Funktion XOR ausgefuehrt.        }

type regs=record case integer of
            1:(ax,bx,cx,dx,bp,si,di,ds,es,flgs: integer);
            2:(al,ah,bl,bh,cl,ch,dl,dh: byte);
          end;
  var reg:regs;
begin
     reg.ah:=$0C;             { Punkt setzen }
     reg.al:=color;           { Farbe        }
     reg.cx:=x;               { x-Position   }
     reg.dx:=y;               { y-Position   }
     intr($10,reg);           { Interrupt 10 }
end;
```

Bit 0...2: Bildschirmspeicherebene
Bit 3...7: Nicht benutzt.

Die Bits 0...2 selektieren die Speicherebene, die vom Prozessor gelesen werden soll. Nur die Werte 0...3 sind zugelassen. Auf Read Mode 1 (Farbvergleich) hat dieses Register keinen Einfluß.

Mode Register

3CEH.05
3CFH

Bit 0...1: Schreibmodus
Bit 2: Test
Bit 3: Lesemodus
Bit 4: Odd/Even
Bit 5: Shift Register Modus
Bit 6...7: Nicht benutzt.

Bit 1 0:

0 0 Schreibmodus 0
Alle im Map Mask Register erlaubten Speicherebenen werden beschrieben, solange nicht für diese Speicherebene(n) Enable Set/Reset selektiert ist (siehe Enable Set/Reset Register).

0 1 Schreibmodus 1
Alle im Map Mask Register erlaubten Speicherebenen werden mit dem Inhalt der Prozessor-Latches geladen.

1 0 Schreibmodus 2
Alle im Map Mask Register erlaubten Speicherebenen und gleichzeitig im Bit Mask Register erlaubten Bits werden mit der Farbe geschrieben, die das Byte, das vom Prozessor kommt, enthält. Diese ist die einzige Operation, mit der es möglich ist, die Farbe direkt zu bestimmen.

1 1 Nicht gültig.

Die logischen Funktionen finden in allen drei Schreibmodi Anwendung.

Bit 2: Test
Ist dieses Bit logisch 1, so werden die Ausgänge der Grafik-Controller hochohmig geschaltet. Dieses Bit wird nur zum Testen der EGA benutzt.

Bit 3: Lesemodus
Beim Lesemodus 0 (Bit 3 = 0) werden die 8 Bit des adressierten Bytes aus der Speicherebene gelesen, die durch das Read Map Select ausgewählt wurde.

Beim Lesemodus 1 (Bit 3 = 1) wird ein Farbvergleich zwischen dem Color Compare Register und dem adressierten Byte durchgeführt (Siehe Color Compare Register).

Bit 4: Ungerade/Gerade
Der Ungerade/Gerade-Adreßmodus (Bit 4 = 1) wird bei Emulation des Color-Grafik-Adapters verwendet. Ebenso, wenn die IBM-EGA-Karte nur mit 64 KByte ausgerüstet ist.

Bit 5: Shift Register Modus
Dieses Bit steuert (wenn gesetzt) den seriellen Datenstrom in Betriebsart 4/5, wenn ein IBM-Farb-Bildschirmgerät angeschlossen ist.

Miscellaneous Register

3CEH.06
3CFH

Bit 0: Grafik Modus
Bit 1: Verkette ungerade/gerade Speicherebenen.
Bit 2...3: Bildschirmspeicher-Adressen
Bit 4...7: Nicht benutzt

Bit 0: Grafik Modus
Ist das Bit gesetzt, so ist der Grafik Modus selektiert.

Bit 1: Verkette ungerade/gerade Speicherebenen
Ist dieses Bit gesetzt, so wird das Bit 0 der Bildschirmspeicher-Adresse dazu benutzt, die gerade, beziehungsweise ungerade Bildschirm-Speicherebene zu adressieren.

Bit
3 2 Bildschirmspeicher-Adressen
0 0 A000H für 128 KByte
0 1 A000H für 64 KByte
1 0 B000H für 32 KByte
1 1 B800H für 32 KByte

Color Don't Care Register

3CEH.07
3CFH

Bit 0: Speicherebene 0 Farbe
Bit 1: Speicherebene 1 Farbe
Bit 2: Speicherebene 2 Farbe
Bit 3: Speicherebene 3 Farbe
Bit 4...7: Nicht benutzt

Dieses Register erlaubt eine Maskierung der Speicherebenen (Farbebenen) beim Lesemodus 1. Ist z. B. das Register mit 0AH geladen, so nehmen am Farbvergleich nur die Speicherebenen 3 und 1 teil. Der Inhalt der Speicherebenen 0 und 2 wird ignoriert.

Bit Mask Register

3CEH.08
3CFH

Bit 0...7: Bit-Maske

Die Bit-Maske wird bei allen Schreiboperationen benutzt. Mit dieser Maske ist es möglich, einzelne oder eine beliebige Anzahl von Bits (Bildpunkte) innerhalb des adressierten Bytes zu selektieren. Ist die Bit-Maske 1, so kann der neue Wert ungehindert in die entsprechende Bitposition der Bildebene geschrieben werden. Ist die Bit-Maske 0, so wird der Wert aus dem Prozessor-Latch-Register in diese Bitposition geschrieben. Will man z. B. nur ein Bit neu schreiben und die anderen Bits in diesem Byte unverändert lassen, so liest man zuerst dieses Byte. Dadurch wird das Prozessor-Latch-Register mit dem Inhalt dieses Bytes geladen. Die nachfolgende Schreiboperation mit der Bit-Maske = 1 schreibt jetzt nur dieses eine Bit neu in das Byte. Der Rest kommt aus dem Prozessor-Latch-Register. Bei der bereits bekannten Prozedur PSET wird diese Lese-/Schreiboperation durch einen einzigen Pascalbefehl erreicht, nämlich:

MEM (egabase: offset) := MEM (egabase: offset);

Ein weiteres Beispiel: Will man den Inhalt eines Bildschirm-Bytes von der Adresse A000:01H nach A000:02 kopieren, so ist folgendes zu tun:

1. Setze das Bit Mask Register = 00H
2. Lese das Byte aus Adresse A000:01H
3. Schreibe das Byte in Adresse A000:02H

Auf diese Weise ist es möglich, 32 Bit innerhalb von zwei Speicherzyklen zu bewegen.

Da alle Register des Grafik-Controllers nur beschrieben, nicht gelesen werden können, ist es wichtig zu wissen, welche Bits sich zu irgend einem Zeitpunkt in den Registern befinden. *Bild 2* zeigt alle Register des Grafik-Controllers und ihren Inhalt, nachdem der jeweilige EGA-Modus (INT10H, AH=0) gesetzt wurde.

Attribute Controller Programmierung

Im Attribute Controller befinden sich im wesentlichen die 16 Farbpaletten Register. Neben diesen Registern gibt es noch

drei weitere Kontrollregister auf dem Attribute Controller, von denen die letzten zwei, das Color Plane Enable Register und das Horizontal Pel Panning Register im Grafikmodus eine besondere Rolle spielen.

Im BIOS der EGA sind für die Programmierung des Attribute Controllers nur zwei Funktionen untergebracht, und zwar INT10H, AH=0BH (nur für den Emulationsmodus) und INT10H, AH=10H. Beide Funktionen kontrollieren nur die Farbpaletten Register. Daß das BIOS keine weiteren Funktionen für den Attribute Controller enthält, hat ähnliche Gründe wie die im Kapitel über den Grafik Controller schon erwähnten.

Attribute Controller Register

Ebenso wie beim Grafik Controller hat auch der Attribute Controller nur sogenannte Write Only Register, deren Inhalt nur geändert, aber nicht gelesen werden kann.

Attribute Address Register

3C0H

Eine Besonderheit bei der Adressierung der Attribute Register ist, daß der Attribute Controller kein Adreßbit für die Selektion zwischen Adreß- und Datenregister hat. Alle Portadressen sind 3C0H. Ein internes Flip-Flop adressiert entweder das Attribute Address Register oder die Datenregister. Dieses Flip-Flop wird durch eine INPUT (IOR) Instruktion entweder auf die Portadresse 3BAH (alle monochromen Modi) oder 3DAH (alle Farb-Modi) so gesetzt, daß das Attribute Address Register selektiert ist.

Bit 0...4: Attribute Adresse
Wird die Adresse in die Bits 0...4 geladen, so wird das Flip-Flop umgeschaltet und es zeigt nun auf das Datenregister. Das Flip-Flop schaltet nun bei jeder OUT Operation hin und her.

Bit 5: Palette-Adressen-Steuerung
Dieses Bit muß 0 sein, wenn ein Palette Register geladen werden soll. Dabei wird gleichzeitig der ganze Bildschirm dunkel gesteuert (eine gute Möglichkeit der Bildschirmsteuerung). Ist das Bit 1, so wird der Bildschirm wieder angeschaltet.

Bit 6...7: Nicht benutzt.

Palette Register 00-0FH

3C0H.00-0FH

Bit 0: Farbe blau
Bit 1: Farbe grün
Bit 2: Farbe rot
Bit 3: Farbe hell blau (monochrome video)
Bit 4: Farbe hell grün (monochrome video hell)
Bit 5: Farbe hell rot
Bit 6...7: Nicht benutzt

Eine logische 1 selektiert die entsprechende Farbe. Um kurzes Aufblitzen des Bildschirmes zu vermeiden, sollten die Palette Register nur während des Bildwechsels geändert werden.

Bit 0...5:
00 schwarz
01 blau
02 grün
03 cyan
04 rot
05 magenta
06 braun
07 weiß
38 dunkel grau
39 hell blau
3A hell grün
3B hell cyan
3C hell rot
3D hell magenta
3E gelb
3F hell weiß

Mode Control Register

3C0H.10H

Bit 0: Grafik/Alphanumerik-Modus
Bit 1: Bildschirm-Typ
Bit 2: Line Grafik Charakter Codes
Bit 3: Erlaube Blinken/Hintergrundhelligkeit
Bit 4...7: Nicht benutzt

Bit 0: Eine logische 0 selektiert den alphanumerischen Modus, eine logische 1 den Grafikmodus.
Bit 1: Eine logische 0 selektiert den Farbbildschirm-Typ, eine logische 1 den monochromen Bildschirm-Typ.
Bit 2: Ist das Bit 0, so hat das neunte Dot-Bit die gleiche Farbe wie die Hintergrundfarbe. Für den IBM-Monochrom-Bildschirm muß dieses Bit 1 sein. Das neunte Dot-Bit hat dann die gleiche Farbe wie das achte.
Bit 3: Eine logische 1 erlaubt Blinken im Alphanumerik- und Grafik-Modus.

Overscan Color Register

3C0H.11H

Bit 0...5: Randfarbe beim IBM-Color-Bildschirm. Farbe wie bei den Palette Registern.

Bit 6..7: Nicht benutzt.
Diese Randfarbe läßt sich nur beim IBM-

Register			Mode of Operation																	
Name	Port	Index	0	1	2	3	4	5	6	7	D	E	F	10	F*	10*	0*	1*	2*	3*
Grphx I Pos	3CC	–	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Grphx II Pos	3CA	–	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
Grphx I II AD	3CE	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Set Reset	3CF	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Enable S/R	3CF	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Color Compare	3CF	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Data Rotate	3CF	03	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Read Map Sel	3CF	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Mode Register	3CF	05	10	10	10	10	30	30	00	10	00	00	10	10	00	00	10	10	10	10
Miscellaneous	3CF	06	0E	0E	0E	0E	0F	0F	0D	0A	05	05	07	07	05	05	0E	0E	0E	0E
Color No Care	3CF	07	00	00	00	00	00	00	00	00	0F	0F	0F	0F	0F	0F	00	00	00	00
Bit Mask	3CF	08	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
*Values for these modes when the IBM Enhanced Color Display is attached																				
*Values for these modes when greater than 64K Graphics Memory is installed																				

Graphics SI Registers

Bild 2. Aus dem Handbuch: Grafics SI Register

Color-Bildschirm selektieren. Bei allen anderen Bildschirmen sollte der Inhalt des Registers auf 0 gesetzt werden.

Color Plane Enable Register

3C0H.12H

Bit 0: Erlaubt Speicherebene 0 (Farbe)
Bit 1: Erlaubt Speicherebene 1 (Farbe)
Bit 2: Erlaubt Speicherebene 2 (Farbe)
Bit 3: Erlaubt Speicherebene 3 (Farbe)
Bit 4...5: Farbe zum Status Register 1
Bit 6...7: Nicht benutzt

Die Bits 0...3 sind für grafische Anwendungen von besonderer Bedeutung. Ist zum Beispiel nur das Bit 0 logisch 1, so wird auch nur der Inhalt der Speicherebene 0 zur Adressierung der Palette Register benutzt.Es wird dann das Palette Register 01 (Farbe blau) adressiert. Alle Bit in dieser Speicherebene werden blau dargestellt. Wird zum Beispiel in jeder Speicherebene ein anderes zweifarbiges Byte abgelegt, so kann hier mit Hilfe des Color Plane Enable Register das Bild ausgewählt werden, das auf dem Bildschirm erscheinen soll. Sind zwei Bilder gleichzeitig selektiert, zum Beispiel Speicherebene 0 und 3, so adressiert ein Bit aus Speicherebene 0 das Palette Register 1 (Farbe blau) und ein Bit aus der Speicherebene 3 das Palette Register 8 (Farbe dunkel grau). Liegen die Bit in beiden Speicherebenen im gleichen Byte und in der gleichen Bitposition, so wird das Palette Register 9 (Farbe hell blau) adressiert. Auf diese Weise lassen sich in einer EGA-Karte mit 256 KByte 8 gleichzeitig gespeicherte zweifarbige Bilder beliebig auswählen. Normalerweise sind alle Speicherebenen erlaubt (0FH), so daß ein 16farbiges Bild erzeugt werden kann.

Register			Mode of Operation																
Name	Port	Index	0	1	2	3	4	5	6	7	D	E	F	10	F‡	10‡	0*	1*	2*	3*
Palette	3C0	0C	14	14	14	14	14	14	17	18	14	14	00	04	00	3C	3C	3C	3C	3C
Palette	3C0	0D	15	15	15	15	15	15	17	18	15	15	18	07	18	3D	3D	3D	3D	3D
Palette	3C0	0E	16	16	16	16	16	16	17	18	16	16	00	00	00	3E	3E	3E	3E	3E
Palette	3C0	0F	17	17	17	17	17	17	18	17	17	00	00	00	3F	3F	3F	3F	3F	3F
Mode Control	3C0	10	08	08	08	08	01	01	01	0E	01	01	0B	0B	0B	01	08	08	08	08
Overscan	3C0	11	00	00	00	00	00	00	00	00	00	00	09	00	00	00	00	00	00	00
Color Plane	3C0	12	0F	0F	0F	0F	03	03	01	0F	0F	0F	05	05	05	0F	0F	0F	0F	0F
Hrz Panning	3C0	13	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
*Values for these modes when the IBM Enhanced Color Display is attached																				
‡Values for these modes when greater than 64K Graphics Memory is installed																				

Attribute Registers (2 of 2)

Bild 4. Das zweite Attributregister

Register			Mode of Operation																
Name	Port	Index	0	1	2	3	4	5	6	7	8	E	F	10	F‡	10‡	0*	1*	2*	3*
Address	3?A	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Palette	3C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Palette	3C0	01	01	01	01	01	13	13	17	08	01	01	08	01	08	01	01	01	01	01
Palette	3C0	02	02	02	02	02	15	15	17	08	02	02	00	00	00	02	02	02	02	02
Palette	3C0	03	03	03	03	03	17	17	17	08	03	03	00	00	00	03	03	03	03	03
Palette	3C0	04	04	04	04	04	02	02	17	08	04	04	18	04	18	04	04	04	04	04
Palette	3C0	05	05	05	05	05	04	04	17	08	05	05	18	07	18	05	05	05	05	05
Palette	3C0	06	06	06	06	06	06	06	17	08	06	06	00	00	00	06	14	14	14	14
Palette	3C0	07	07	07	07	07	07	07	17	08	07	07	00	00	00	07	07	07	07	07
Palette	3C0	08	10	10	10	10	10	10	17	10	10	10	00	00	00	38	38	38	38	38
Palette	3C0	09	11	11	11	11	11	11	17	18	11	11	08	01	08	39	39	39	39	39
Palette	3C0	0A	12	12	12	12	12	12	17	18	12	12	00	00	00	3A	3A	3A	3A	3A
Palette	3C0	0B	13	13	13	13	13	13	17	18	13	13	00	00	00	3B	3B	3B	3B	3B
? = B in monochrome modes ? = D in color modes																				
*Values for these modes when the IBM Enhanced Color Display is attached																				
‡Values for these modes when greater than 64K Graphics Memory is installed																				

Attribute Registers (1 of 2)

Bild 3. Das erste Attributregister

Bit 4..5: Farbe zum Status 1 Register. Diese Bits werden nur beim Power-On-Test benötigt.

Horizontal Pel Panning

3C0H.13H

Bit 0..3: Horizontale Verschiebung
Diese 4 Bit erlauben die Verschiebung des Bildes horizontal nach links um einzelne Bildpunkte sowohl im Alphanumerik- als auch im Grafik-Modus.

Monochrome Modus (alphanum.)
8,0,1,2,3,4,5,6,7
Alle anderen Modi
0,1,2,3,4,5,6,7

Dieses Register erlaubt in Verbindung mit dem CRTC das sogenannte horizontale Scrolling.

Da alle Register des Attribute Controllers sich nur schreiben, aber nicht lesen lassen, ist es wichtig zu wissen, welchen Inhalt die Register zu jedem Zeitpunkt haben. Die *Bilder* 3 und 4 zeigen alle Register des Attribute Controllers und ihren Inhalt direkt nachdem der jeweilige EGA-Modus (INT10H, AH=0) gesetzt wurde. Alle anderen EGA-Register sollten aus Sicherheitsgründen nicht angefaßt werden. Zum Beispiel ist die Hochspannung eines Videomonitors von der Zeilenfrequenz abhängig. Eine falsch programmierte EGA könnte den Bildschirm zerstören.

Literatur

Die Bilder 2...4 sind ein genehmigter Nachdruck aus dem Handbuch zum IBM-Enhanced-Graphics-Adapter.

Frank Wolter

Sprites für den Apple-II

Obwohl der Apple von Haus aus über sogenannte „Shapes" verfügt, soll hier eine schnellere und speicherfreundlichere Methode zur Darstellung von Grafiksymbolen vorgestellt werden.

Jeder, der schon einmal versucht hat, Shapes zu entwerfen, wird wohl bestätigen können, daß dieses Vorhaben ohne ein Programm, das die Codierung der Shapes vornimmt, so gut wie unmöglich ist oder zumindest einen ganzen Batzen Zeit in Anspruch nimmt. Zudem sind solche Shapes in Programmen, in denen es auf schnelle Grafik ankommt, oft nicht einzusetzen, da die Ausführung der Vektorkommandos auch für einen Mikroprozessor eine zeitraubende Angelegenheit ist. Hinzu kommt noch, daß die Vektortabellen der Shapes sich im Speicher schnell auf ungeahnte Größen aufblähen, wenn einige größere Objekte definiert wurden.
Eine elegantere Möglichkeit bieten Sprites, auch Blockshapes genannt. Sie bestehen im Gegensatz zu den Shapes nicht aus Kommandocodes, sondern direkt aus dem Bitmuster der Grafik, so wie es später auf dem Bildschirm ausgegeben wird. Daß es schneller geht, eine bestehende Grafik von einem Bereich des RAMs in einen anderen zu kopieren, als Vektorkommandos zu interpretieren und auszuführen, wird jedem einleuchten.

Der Aufbau der Sprites

Das Programm orientiert sich anhand eines Zeigers, wo es die Sprites im Speicher finden kann. Dieser Zeiger kann vom Benutzer bequem per Basic-Befehl definiert werden. So besteht die Möglichkeit, die Tabelle an einer beliebigen Stelle im Speicher des Rechners zu plazieren. Natürlich ist hierbei darauf zu achten, daß die Daten nicht mit reservierten Speicherbereichen (Basic-Bereich, Sprite-Routinen, DOS usw.) in Konflikt kommen.
Wie der Zeiger beeinflußt werden kann, wird später gezeigt. An der Stelle, auf die der Zeiger deutet, befinden sich wiederum zwei Bytes, die auf das nächste Sprite hinweisen. Dies soll ein schnelles Auffinden ähnlich wie bei den Basic-Zeilen ermöglichen. Das folgende Byte gibt die vom Benutzer festgelegte Nummer des Sprites an. Sie darf im Bereich von 1 bis 255 liegen und kann vom Programmierer frei gewählt werden.
Als nächstes folgen einige Informationen, auf die der Benutzer allerdings nur indirekt Einfluß hat. Es handelt sich um Angaben zur Darstellung der Sprites (*Bild 1*) die intern von den Sprite-Routinen verwaltet werden. Sie beinhalten: Die Breite in Bytes, die erforderlich ist, um alle Punkte des Objektes in der Breite zu erfassen. In der Sprite-Tabelle werden jeweils in einem Byte acht Punkte der Grafik abgelegt. Hat eine Grafik in der Breite z. B. elf Punkte, so sind zwei Bytes zur Darstellung der Punkte erforderlich; acht Punkte in dem einen Byte und drei Bits des zweiten Bytes werden benutzt, die anderen sind auf Null gesetzt.
Gefolgt wird dieses Byte von der Gesamtlänge in Bytes, die zur Darstellung des Musters erforderlich ist. Hier wird eine Einschränkung für den Benutzer gemacht. Es dürfen maximal 255 Bytes für ein Sprite verwendet werden. Diese Länge wird vom Programm aus kontrolliert, und gegebenenfalls wird der Programmierer bei Mißachtung dieser Regel gewarnt.
Abschließend folgen dann endlich die Bytes, aus denen das Sprite besteht, wobei das Objekt zeilenweise im Speicher abgelegt ist.
Mit dem Befehl „&Clear (Ausdruck)" können alle vorhandenen Sprite-Definitionen gelöscht werden, gleichzeitig wird der Zeiger auf die Sprites neu festgelegt.

```
Sprite # 1
Zeiger auf Sprites --->     1.  Zeiger auf nächstes Sprite -+
                            2.  High-Byte vom Zeiger       -+
                                                            I
                            3.  Kennummer des Sprites       I
                                                            I
                            4.  Breite in Bytes             I
                            5.  Anzahl der folgenden Bytes  I
                                                            I
                            6.  Sprite als Bitmuster        I
                                        ''                  I
                                        ''                  I
                          max. 255. Ende des Sprites        I
                                                            I
Sprite # 2                  1.  $00                 <-------+
                            2.  $00
                            3.  $00
```

Bild 1. Aufbau eines Sprite im Speicher: Drei Null-Bytes kennzeichnen das Ende der Tabelle

Die Eingabe von Sprites

Eine Definition (*Bild 2*) beginnt mit dem Kommando „&Def (Ausdruck)", wobei der Ausdruck die Nummer des Sprites angibt (1...255). Jetzt folgt das eigentliche Sprite auf eine übersichtliche und einfache Weise:

Beispiel:

```
&Def ..X..
&Def XXXXX
&Def ..X..
```

Dabei erscheint später auf dem Bildschirm für jedes X ein heller Punkt. Abschließend wird das Ende der Definition mit „&Def End" gekennzeichnet. So, und nun endlich kann das Sprite in der gewohnten Weise, wie ein Shape auf den Bildschirm gezaubert werden. Hierzu dienen die Befehle
&DRAW (Nummer) AT (X-Position), (Y-Position)
&XDRAW (Nummer) AT (X-Position), (Y-Position)
&DRAW zeichnet das Sprite über schon bestehende Grafiken, wohingegen &XDRAW das Sprite invertierend mit dem Hintergrund ausgibt, was dazu benutzt werden kann, um ein Sprite wieder vom Bildschirm zu löschen. Jeder Ausdruck, der hier in Klammern geschrieben wurde, kann eine Zahl, eine Variable oder auch ein Ausdruck sein.

Kollisionen feststellen

Manchmal ist es sinnvoll, feststellen zu können, ob das gerade gezeichnete Sprite mit einem Objekt auf dem Bildschirm (gesetzte weiße Punkte) kollidiert ist. Dies ist nach dem Draw-Kommando

durch einen einfachen Test der Speicherstelle 255, z. B. mit Koll=Peek (255) zu ermitteln. Ergibt das den Wert Null, so wurde das Sprite auf schwarzem Grund ausgegeben, ansonsten wurde ein weißer Punkt überschrieben.

Fehlermeldungen

Sprite too long wird ausgegeben, wenn der Gesamtumfang des Sprites 255 Bytes überschreitet. Hier hilft nur, das Sprite in zwei kleinere zu unterteilen, und diese dann nacheinander auf dem Bildschirm auszugeben.

Duplicate Spritenumber: Sie haben versucht, einem Sprite eine Nummer zuzuweisen, die im Speicher schon existiert. Benutzen Sie eine andere Nummer, oder führen sie ein &Clear aus, um alte Definitionen aus dem Speicher zu löschen.

Undefined Sprite erscheint, falls Sie ein Draw- bzw. XDraw-Kommando ausführen und dabei ein Sprite aufrufen, das noch nicht definiert wurde.

End without Def erhalten Sie beim Aufruf des &Def-End-Befehls, bevor eine Definition durch &Def (Nummer) eingeleitet wurde. Entfernen Sie den überflüssigen Befehl.

Missing End: Sie haben eine Definition begonnen, und ohne diese mit dem End-Befehl abzuschließen, wurde versucht, ein Sprite mittels Draw-Kommandos zu zeichnen oder ein weiteres Sprite zu definieren. Die Sprites werden nur zur Benutzung freigegeben, wenn alle Definitionen vorschriftsmäßig abgeschlossen sind. Fügen Sie den fehlenden End-Befehl ein.

Missing Spritenumber wird gemeldet, falls sofort mit der Definition eines Sprites (&Def ...X..X..X.) begonnen wurde, ohne die Startzeile, in der dem Sprite eine Nummer zugewiesen wird, ausgeführt zu haben. Weisen Sie dem Sprite zu Anfang eine Nummer zu.

```
9200- A9 4C 8D F5 03 A9 AB 8D +=045B
9208- F6 03 A9 93 8D F7 03 A0 +=045C
9210- 00 84 1E A9 60 85 1F 98 +=02E7
9218- 91 1E C8 91 1E C8 91 1E +=039D
9220- A9 29 85 73 A9 92 85 73 +=03FD
9228- 60 00 00 00 00 00 00 00 +=0060
9230- 00 00 00 A5 1E 85 CE A5 +=02BB
9238- 1F 85 CF A0 00 60 A0 00 +=0313
9240- B1 CE AA C8 B1 CE 85 CF +=05C4
9248- 86 CE D0 10 8A D0 0D 68 +=0403
9250- 68 AD 32 92 F0 03 4C 6F +=0387
9258- 95 4C 8A 95 60 20 B1 00 +=0331
9260- 20 67 DD 20 52 E7 A9 00 +=0366
9268- 8D 30 92 8D 31 92 A8 91 +=03D8
9270- 50 C8 91 50 A5 50 85 1E +=0391
9278- A5 51 85 1F 60 A2 00 8E +=032A
9280- 2D 92 8E 2E 92 86 FF 86 +=0418
9288- FA A5 30 29 7F C9 01 F0 +=0431
9290- 0C 4A 38 66 FA EE 2D 92 +=039B
9298- 6A 90 F7 46 FA 60 AA 25 +=0460
92A0- FA 8D 2F 92 8A 18 AE 2D +=03C5
92A8- 92 F0 04 0A CA D0 FC 0D +=0433
92B0- 2E 92 F0 12 AA 31 26 29 +=02EC
92B8- 7F F0 04 A9 01 85 FF 8A +=042B
92C0- 11 26 29 7F 91 26 AD 2F +=0272
92C8- 92 AE 2D 92 F0 07 E8 E8 +=04C6
92D0- 18 2A CA D0 FC 8D 2E 92 +=0425
92D8- 60 A9 51 8D C0 92 4C ED +=0472
92E0- 92 20 B1 00 20 B9 F6 60 +=0392
92E8- A9 11 8D C0 92 AD 31 92 +=0409
92F0- F0 03 4C 8F 95 20 B1 00 +=0334
92F8- 20 67 DD 20 52 E7 A5 51 +=03B3
9300- D0 0D A5 50 F0 09 8D 29 +=0381
9308- 92 20 33 92 4C 15 93 4C +=02B7
9310- 99 E1 20 3E 92 A0 02 B1 +=03BD
9318- CE CD 29 92 D0 F4 C8 B1 +=0593
9320- CE 85 FD F0 BC C8 B1 CE +=0643
9328- 85 FC A9 05 85 FB 20 B1 +=0480
9330- 00 20 B9 F6 8E 2A 92 8C +=03A5
9338- 2B 92 8D 2C 92 20 11 F4 +=032D
9340- 20 7D 92 4C 59 93 AE 2A +=033F
9348- 92 AC 2B 92 EE 2C 92 AD +=0454
9350- 2C 92 C9 C0 F0 54 20 11 +=03BC
9358- F4 A5 FD 85 FE 8C 29 92 +=0560
9360- A4 FB B1 CE AC 29 92 20 +=04A5
9368- 9E 92 E6 FB C6 FC F0 30 +=05F3
9370- C6 FE F0 17 C8 C0 28 90 +=050B
9378- E4 E6 FB C6 FC F0 2B C6 +=0668
9380- FE D0 F1 A9 00 8D 2E 92 +=04B5
9388- 4C 46 93 C8 C0 28 B0 08 +=038D
9390- A9 00 20 9E 92 4C 46 93 +=031E
9398- A9 00 8D 2E 92 4C 46 93 +=031B
93A0- A9 00 C8 C0 28 B0 03 20 +=032C
93A8- 9E 92 60 A0 00 8C 32 92 +=0380
93B0- B1 B8 C9 BD D0 03 4C 5D +=046B
93B8- 92 C9 B8 D0 03 4C D1 93 +=0496
93C0- C9 94 D0 03 4C E8 92 C9 +=04BF
93C8- 95 D0 03 4C D9 92 4C C9 +=0434
93D0- DE 20 B1 00 EE 32 92 A0 +=0401
93D8- 00 B1 B8 C9 80 F0 22 C9 +=048D
93E0- 2E D0 03 4C 84 94 C9 58 +=0386
93E8- D0 03 4C 84 94 20 67 DD +=039B
93F0- 20 52 E7 A5 51 D0 07 A5 +=03CB
93F8- 50 F0 03 4C 32 94 4C 99 +=033A
9400- E1 A4 FB AD 31 92 D0 03 +=04C3
9408- 4C 65 95 CE 31 92 A9 00 +=0380
9410- 91 CE C8 91 CE C8 91 CE +=05AD
9418- 88 88 18 98 65 CE 48 A5 +=03E0
9420- CF 69 00 A0 01 91 CE 88 +=03C0
9428- 68 91 CE EE 30 92 20 B1 +=0448
9430- 00 60 AD 31 92 F0 03 4C +=030F
9438- 8F 95 20 33 92 EE 31 92 +=03BA
9440- AC 30 92 8C 29 92 F0 1B +=03C0
9448- A0 02 B1 CE C5 50 F0 0B +=0431
9450- 20 3E 92 A0 02 B1 CE C5 +=03D6
9458- 50 D0 03 4C 6F 95 CE 29 +=036A
9460- 92 D0 ED A0 02 A5 50 91 +=0477
9468- CE A9 00 85 66 85 EB 85 +=0457
9470- EC C8 91 CE C8 91 CE A0 +=05DA
9478- 05 84 FB 60 68 68 4C 6A +=036A
9480- 95 20 B1 00 AD 31 92 C9 +=039F
9488- 01 D0 68 A0 00 B1 B8 C9 +=040B
9490- 58 F0 05 C9 2E D0 43 18 +=036F
9498- 66 66 E6 EB A5 EB C9 07 +=04FD
94A0- D0 DF A4 FB A5 66 4A 91 +=0534
94A8- CE E6 FB F0 CF A9 00 85 +=059C
94B0- 66 85 EB A0 04 B1 CE AA +=04A3
94B8- E8 BA 91 CE 88 E6 EC A5 +=05D0
94C0- EC 91 CE 20 B1 00 A0 00 +=03BC
94C8- B1 B8 C9 2E F0 04 C9 58 +=0475
94D0- D0 03 4C 84 94 A9 00 85 +=0365
94D8- EC 60 A4 EB C0 07 F0 06 +=0498
94E0- 46 66 C8 4C DC 94 C6 B8 +=04AE
94E8- A5 B8 C9 FF D0 02 C6 B9 +=0576
94F0- 4C A2 94 A0 56 4C 76 95 +=03CF
94F8- BF C5 CE C4 A0 D7 C9 D4 +=062A
9500- C8 CF D5 D4 A0 C4 C5 C6 +=062F
9508- 00 BF D3 D0 D2 C9 D4 C5 +=0596
9510- A0 D4 CF CF A0 CC CF CE +=061B
9518- C7 00 BF C4 D5 D0 CC C9 +=0584
9520- C3 C1 D4 C5 A0 D3 D0 D2 +=0632
9528- C9 D4 C5 CE D5 CD C2 C5 +=0659
9530- D2 00 BF D5 CE C4 C5 C6 +=0583
9538- A7 C4 A0 D3 D0 D2 C9 D4 +=061D
9540- C5 00 BF CD C9 D3 D3 C9 +=0589
9548- CE C7 A0 C5 CE C4 00 BF +=054B
9550- CD C9 D3 D3 C9 CE C7 A0 +=063A
9558- D3 D0 D2 C9 D4 C5 CE D5 +=067A
9560- CD C2 C5 D2 00 A0 FF 4C +=0511
9568- 76 95 A0 10 4C 76 95 A0 +=03B2
9570- 21 A9 00 8D 31 92 20 8E +=02C8
9578- FD C8 B9 F8 94 F0 06 20 +=0520
9580- ED FD 4C 79 95 68 68 4C +=0460
9588- 2D D4 A0 39 4C 76 95 A0 +=03D1
9590- 49 4C 76 95 +=01A0
```

Bild 2. So definiert man ein Sprite. Die zugehörige Tabelle, die im Speicher angelegt wird, ist im Bild unten zu sehen

Hinweise

Wird nach dem Starten des Maschinenprogramms gleich mit der Definition von Sprites begonnen, so steht der Zeiger auf die Sprites bei $6000, also hinter der zweiten HGR-Seite.
Beachten Sie, daß folgendes möglich ist:

```
&Def 1 : &DefXXXXX
For I = 1 to 10
&Def X...X
Next
&Def XXXXX
&Def End
```

Dieser kleine Trick kann einem manchmal viel Arbeit ersparen.
Achten Sie darauf, daß in allen Definitionszeilen des Sprites die Anzahl der Punkte („.“ oder „X“) gleich ist und keine Schwankungen in der Breite des Sprites vorliegen. Ansonsten könnte es passieren, daß die Sprites nicht richtig in der Tabelle abgelegt werden.

```
Die Definition des Sprites:

100 &CLEAR 6*256*16          150 &DEF XXXXX
110 &DEF 1                   160 &DEF X...X
120 &DEF ..X..               170 &DEF END
130 &DEF .X.X.               200 HGR 2
140 &DEF X...X               210 &DRAW 1 AT 10,10

Die Tabelle im Speicher:

Adresse  Pointer  Nummer Breite Bytes Bitmuster

6000:    0A 60    01     01     05    04 0A 11 1F 11
600A:    00 00 00 (=Ende)
```

Bild 3. Das Programm ist mit BSAVE SPRITES, A$9200, L$394 abzuspeichern und vor seiner Verwendung mit BRUN SPRITES zu laden [1]

Literatur

[1] *Feichtinger, Herwig:* Prüfsummenprogramm für den Apple-II. mc 1984, Heft 6, Seite 64...65.

Wolfgang Schrader

TURBO-Takt-umschaltung

Besitzer eines „XT-Kompatiblen" mit 8-MHz-Turbo-Board kennen das Problem: Das Umschalten zwischen den beiden Taktfrequenzen 4,77 MHz und 8 MHz funktioniert nur so lange, bis KEYBGR.COM geladen und gestartet wird. Die folgenden Programme schaffen Abhilfe.

```
A>debug
-N KEYBGR.COM
-L
-
-D 1FB L15
6C94:01FØ                                       3C 53 75 0E C7              <Su.G
6C94:0200  06 72 00 34 12 EA 00 00-FF FF E9 0F 01 E9 15 01   .r.4.j....i..i..
-
-E 20E     0C
-E 31D     4A 75 05 E9 D1 03 90 90
-E 6F4     86 C4 B0 0B EA AF E5 00 F0 90 90 90
-E 71F     FD
-
-W
Schreiben von 0625 Bytes
-Q

A>KEYBGR

A>
```

Bild 1. So wird der Tastatur-Treiber KEYBGR.COM mit Debug geändert

```
umsch   segment
        assume  cs:umsch

        org     0100h           ;Programm soll ein .COM-File werden

start   proc    near

        push    ax
        push    bx
        push    cx

        in      al,61h          ;alte Taktfrequenz lesen
        xor     al,0ch          ;neue Portbeschreibung bilden (D2, D3)

        out     61h,al          ;Umschaltung 4/8 MHz

        mov     ah,01h          ;Funktionsaufruf für INT 10h
        mov     cx,0607h        ;Cursorlinien:  6 - 7  ( 4 MHz - Takt )
        and     al,0ch          ;4 oder 8 MHz geschaltet ?
        jz      s1

        mov     ch,00h          ;Cursorlinien:  0 - 7  ( 8 MHz - Takt )

s1:     int     10h             ;mit ah=1 -> Cursorgröße einstellen

        pop     cx
        pop     bx
        pop     ax

        int     20h             ;Programmende

start   endp
umsch   ends
        end start
```

Bild 2. Diese Routine erledigt die Umschaltung gleich beim Booten

Die Umschaltung der beiden Taktfrequenzen erfolgt beim TURBO-Board über die Tastenkombination Ctrl – Alt – Minus. Diese Tastenkombination wird normalerweise von der ROM-BIOS-Tastaturroutine abgefangen und „erledigt". Andere Beispiele für dieses Verfahren sind Ctrl – NumLock, PrtSc, Ctrl – Alt – Del und Ctrl – Break.

Bei der Installation der deutschsprachigen Tastaturroutine (KEYBGR.COM) wird der Vektor für den INT 09H so geändert, daß er nicht mehr in die ROM-BIOS-Tastaturroutine zeigt, sondern in die KEYBGR-Tastaturroutine. Soll die Taktumschaltung auch nach dem Laden von KEYBGR.COM funktionieren, muß eine Abfrage auf Ctrl – Alt – Minus und eine entsprechende Service-Routine dafür eingebaut werden. Bei der Änderung wird davon ausgegangen, daß im KEYBGR.COM eine Abfrage für die Tastenkombination Ctrl – Alt – Del (Warmstart) existieren muß. Das „alte" KEYBGR prüft zuerst Ctrl – Alt und unterscheidet danach die Fälle „Del", „F1" und „F2". Außerdem wird die Tastenkombination Alt – SPACE geprüft (KEYBGR macht daraus „SPACE").

Um in das vorhandene KEYBGR.COM die zusätzliche Abfrage für Ctrl – Alt – Minus einbauen zu können, wird die wohl selten benutzte Abfragesequenz für Alt – SPACE geändert. An dieser Stelle wird nunmehr das AL-Register auf das Byte 4AH getestet. Der Wert 4AH entspricht dem SCAN-Code für die Minus-Taste rechts neben dem Nummernblock. Verläuft dieser Test erfolgreich, wird nach einer Umbelegung des AX-Registers in die eigentliche Umschaltroutine im ROM-BIOS verzweigt. Die Änderung des AX-Registers ist nötig, weil die ROM-BIOS-Tastaturroutine die Register anders belegt als die KEYBGR-Routine. Die gesamte Programmsequenz der Umbelegung des AX-Registers und des FAR JUMP ins ROM-BIOS ist 9 Bytes lang und paßt noch in den nichtbenutzten Raum (12 Bytes) am Ende des KEYBGR.COM.

Alle angegebenen Änderungen wurden an der KEYBGR.COM-Version mit der Länge 1573 Bytes vorgenommen. *Bild 1* zeigt die komplette Änderungsanleitung mit Hilfe des Debuggers. Nach dem Laden des Debuggers (A>DEBUG) wird der Name des zu „behandelnden" Files, in diesem Fall also KEYBGR.COM eingegeben (–N KEYBGR.COM) und mit –L (Load) geladen. Mit –D 1FB L15 (Dump) wird ein 21 Byte langes Hex-Dump ab Adresse 01FBH ausgegeben. Falls Ihr Hex-Dump etwas anderes ergibt, können die Änderungen nicht direkt durchge-

führt werden. In diesem Fall müssen Sie wohl oder übel nach der Anweisung CMP AL,53 (= 3C 53) suchen und die entsprechenden Adressen ändern. Mit –E 20E 0C (Enter) wird das Byte 0CH an die Adresse 020EH geschrieben. Die nachfolgenden Anweisungen ändern die Bytes ab Adresse 031DH, 06F4H und 071FH. Mit der Anweisung –W (Write) wird das geänderte KEYBGR.COM-File auf die Diskette geschrieben. Das alte KEYBGR.COM wird dabei natürlich überschrieben! Mit –Q (Quit) wird der Debugger verlassen und mit A>KEYBGR wird die „neue" Tastaturroutine eingebaut. Jetzt sollte die Taktumschaltung über die Tastenkombination Ctrl – Alt – Minus und über Alt – Minus eigentlich funktionieren.
Das nächste Programm schaltet die 8-MHz-Taktung gleich beim Booten des Systems ein. Benötigt wird hierfür ein kleines Programm, das die Bits D3 und D2 des Ports 61H bedienen kann.
Über diese beiden Bits, die in „grauer Vorzeit" u. a. den Kassettenrecordermotor steuerten, wird beim TURBO-Board die Taktfrequenz hardwaremäßig umgeschaltet. *Bild 2* zeigt ein Assemblerprogramm, mit dem man zwischen den beiden Taktfrequenzen hin- und herschalten kann.
Nach der Sicherung aller benutzten Register wird der bisherige Zustand des Ports 61H gelesen (Port 61H entspricht Port B des 8255). Diese Information ist sehr wichtig und muß z. T. wieder zurückgeschrieben werden, da sonst z. B. die Tastatur vollständig blockiert werden könnte. Mit der nachfolgenden Exclusiv-ODER-Verknüpfung werden die beiden Bits D3 und D2 in den jeweils anderen Zustand gesetzt. Danach erfolgt die eigentliche Umschaltung (OUT 61H,AL) und die Änderung der Cursorgröße in Abhängigkeit der aktuellen Taktgeschwindigkeit über den Funktionsaufruf 01H des INT 10H. Das Programm kann direkt durch AUTOEXEC.BAT gestartet werden.

Eingabe unter Turbo-Pascal

Wie man die Eingabe von Integer- und Real-Werten in Turbo-Pascal-Programmen verbessern kann, wurde in mc 7/1986 ab Seite 89 beschrieben. Manchmal stellt sich jedoch das Problem, daß nur bestimmte Zeichen als Eingabe zulässig sind. Dann hilft das vorliegende Programm (*Bild*).

```
(*************************************************************************)
(* FUNCTION LIESZCHN   Liest ein Zeichen von der Tastatur u. vergleicht es  *)
(*                     mit der vorgegebenen ZeiCHNMENGE.                    *)
(* Syntax :     ZCHN := LIESZCHN([zchn1,zchn2,..,zchnx]);                   *)
(*************************************************************************)

    Function lieszchn(zchnmenge : setofchar) : char;
    Var ch : char;
        ok : boolean;
    Begin
      repeat
        read(kbd,ch);
        ok := ch in zchnmenge;
        if not ok then write(chr(7));
      until ch in zchnmenge;
      write(ch);
      lieszchn := ch
    End; (* of lieszchn *)

(*************************************************************************)
(* PROCEDURE LIESWORT Liest einen String von der Tastatur u. vergleicht ihn *)
(*                    mit der vorgegebenen ZeiCHNMENGE.                    *)
(*  Syntax :     LIESWORT([zchn1,zchn2,..,zchnx],lnge,var);                 *)
(*************************************************************************)

    Procedure lieswort(zchnmenge : setofchar ; maxlaenge : integer;
                       var str : hilftext);
    Var str1    : string[1];
        strtemp : string[80];
    Begin
      zchnmenge := zchnmenge;
      str1 := ' ';
      strtemp := '';
      repeat
        if length(strtemp)=0 then str1[1] := lieszchn(zchnmenge+[chr(13)])
           else if length(strtemp)=maxlaenge
                then str1[1] := lieszchn([chr(13),chr(8)])
                else str1[1] := lieszchn(zchnmenge+[chr(13),chr(8)]);
        if str1[1] in zchnmenge then strtemp := concat(strtemp,str1)
           else if str1[1] = chr(8) then begin
                                           write(' ');
                                           write(chr(8));
                                           delete(strtemp,length(strtemp),1)
                                         end
      until str1[1] = chr(13);
      if length(strtemp)<>0 then str := strtemp
       else write(str)
    End; (* of lieswort *)

A>
```

Die Funktion Lieszchn holt ein Zeichen, die Prozedur Lieswort eine Zeichenkette von der Tastatur. Ungültige Zeichen werden akustisch angezeigt und nicht übernommen

Im Hauptdeklarationsteil müssen dazu folgende Vereinbarungen getroffen werden:

Type SetOfChar = set of char;
Hilftext = string(80);
Diese Definition genügt. Um die Funktion „lieszchn" in eigenen Programmen verwenden zu können, muß mindestens noch eine Char-Variable vereinbart werden, z. B.
VAR zchn : char;
Sollen nun nur bestimmte Zeichen eingelesen werden, so wird die dazugehörige Maske wie folgt definiert:
zchn := lieszchn(['a','b']);
Es wird nun solange gewartet, bis entweder ein a oder ein b eingegeben wird. Bei Eingabe eines anderen Zeichens wird ein Chr(7) ausgegeben, also das Bell-Zeichen. Die Variable zchn kann nun weiterverwendet werden, etwa in
case zchn of
...
end;

Die Definition der Prozedur „lieswort", die eine komplette maskierte Zeichenkette holt, ist etwas umfangreicher. Hierzu muß zunächst ein String vereinbart werden, z.B.
VAR hilfstr : hilftext;
Hilftext wurde ja schon definiert. Soll nun „lieswort" einen Namen einlesen, in dem nur kleine Buchstaben vorkommen sollen, so kann man die entsprechende Maske wie folgt definieren:
lieswort(['a'..'z',' ','-'],20,hilfstr);
Damit darf Hilfstr eine maximale Länge von 20 Zeichen annehmen. Wird diese Länge überschritten oder wird ein nicht der Maske entsprechendes Zeichen eingegeben, so wird das wieder akustisch durch Chr(7) angezeigt. Zulässig sind in diesem Fall alle kleinen Buchstaben, das Leer- und das Minuszeichen.

Frank Albuschat

Stefan Wurdack

Ein Hex-Monitor für den mc-68-ECB

Der Monitor wurde geschrieben, damit Sie einen direkten Kontakt mit dem 68000 im System bekommen. Er verfügt über sämtliche gängige Monitor-Befehle (ähnlich dem CP/M-Debugger DDTZ). Mit diesen Kommandos ist es möglich, selbstgeschriebene Programme unter Monitorkontrolle ablaufen zu lassen und von ihren Fehlern zu befreien. Auch die Installation eines Betriebssystems erfordert als Hilfsmittel einen Monitor.

Das Monitorprogramm wurde modular aufgebaut, um Erweiterungen und die Verwendung von bereits vorhandenen Unterprogrammen durch Anwenderprogramme so einfach wie möglich zu machen. Der Monitor umfaßt etwa 4 KByte Maschinencode und benötigt ungefähr 1 KByte RAM-Speicher. *Tabelle 1* zeigt den Befehlssatz.

Sämtliche Adreß- (aaaaaa bzw. bbbbbb) und Dateneingaben (dd) erfolgen in Hex. Dabei müssen nur die notwendigen Stellen eingegeben werden. ASCII-Zeichen oder Zeichenfolgen werden durch Einschluß in einfache Anführungszeichen gekennzeichnet. Fehleingaben können mit ⟨BS⟩ (Crtl-H) schrittweise korrigiert werden. Die gesamte Eingabe wird durch Crtl-X gelöscht.

Tabelle 1: Die Monitorbefehle

B	Breakpoints verwalten
D	Memorydump
E	Exit zu CP/M
F	FCB für Datei-Lese- und Schreibkommando aufbauen
G	Go
M	Daten im Speicher verschieben
P	Drucker der Bildschirmausgabe parallel schalten
Q	nach einer bestimmten Bytefolge im Speicher suchen
R	Datei von Floppy lesen
S	Speicher byteweise mit Daten füllen
T	Programm schrittweise ausführen
W	Datei auf Floppy schreiben
X	Registeranzeige
Z	Speicher mit Bitmuster füllen

Die Befehle im Einzelnen

B: Anzeigen sämtlicher Breakpoints.

B aaaaaa: Breakpoint auf die Adresse $aaaaaa setzen. Es können bis zu 10 verschiedene Breakpoints gesetzt werden. Die Breakpoints setzt der Monitor erst bei Ausführung eines GO-Kommandos ein, so daß sie durch einen Memorydump an den entsprechenden Stellen nicht zu sehen sind. Als Breakpoint wird der Software-Interrupt „Trap 14" mit dem Opcode $4e4e verwendet. Nachdem die CPU auf diesen Befehl gestoßen ist, übernimmt der Monitor die Kontrolle und zeigt alle Registerinhalte an. Soll das Programm dann weiterlaufen, so ist es nicht notwendig einen Trace-Befehl zur Überwindung des Breakpoints einzugeben. Vielmehr wird der Befehl, der an der Adresse des letzten Breakpoints liegt, im Einzelschrittmodus ausgeführt. Erst dann setzt der Monitor die Breakpoint-Opcodes wieder in das Programm ein, so daß es dann bis zum nächsten Breakpoint in Echtzeit abläuft.

BX: Löscht den ersten Breakpoint aus der Tabelle.

BX N: Löscht den n-ten Breakpoint aus der Tabelle.

D: Memorydump. Befehl zum Auflisten des Speicherinhalts in Hexadezimal- und ASCII-Darstellung.

D: Memorydump ab der Adresse, bei der der letzte Dumpbefehl endete.

D aaaaaa: $ 80 Byte langer Speicherauszug ab Adresse $aaaaaa.

D aaaaaa bbbbbb: Dump von $aaaaaa bis $bbbbbb. Sämtliche Dumpfunktionen können an beliebiger Stelle durch Eingabe irgendeines Zeichens unterbrochen werden.

E: Verlassen des Monitors und Rückkehr ins 8-Bit-CP/M. Der 68000 springt wieder in die RAM-Floppy-Treiberroutinen.

Fd:nnnnnnnn.eee: Vor dem Lesen oder Schreiben einer Datei mit der Read oder Write Funktion muß der FCB (File Control Block) mit dem „F"-Kommando aufgebaut werden. Dabei kann der Laufwerksname (d:) zusätzlich zum Dateinamen (nnnnnnnn.eee) angegeben werden. Der Zugriff auf Dateien die sich auf der RAM-Floppy befinden ist leider nicht möglich. Der Grund dafür ist, daß der 68000 in die RAM-Floppy-Treiberroutinen springen müßte, nachdem er das Lese- oder Schreibkommando abgesetzt hat und von dort auch nach erfolgreicher Datenübertragung wieder zurückkommen sollte, um die Daten, die er gerade an den Z80 geschickt hat, empfangen zu können. Eine einfache Lösung dieses Problems ließe sich am besten mit einer Interruptsteuerung erreichen. Dies wäre leicht möglich, wenn man z. B. Bit 7 des 68000 Steuerlatches mit einem Interrupteingang verbände und damit im geeigneten Moment das Anspringen einer Serviceroutine erzwingen könnte. Da der Datenaustausch zwischen den beiden Prozessoren dadurch aber erheblich komplizierter würde, wurde in der vorliegenden Monitorversion auf den RAM-Floppy-Zugriff verzichtet.

G: Programmausführung ab dem aktuellen Programmzählerstand. Die CPU-Register werden dabei aktuell, d. h. so wie sie der „X"-Befehl zeigt eingesetzt.

G aaaaaa: Programmausführung ab PC = $aaaaaa.

M aaaaaa bbbbbb cccccc: Daten von $aaaaaa bis $bbbbbb nach $cccccc verschieben.

P: Der Drucker wird der Bildschirmausgabe parallelgeschaltet, d. h. eine Kopie sämtlicher Bildschirmausgaben wird zum Drucker gesendet. Die Funktion kann durch nochmalige Eingabe von „P" wieder abgeschaltet werden.

Q aaaaaa dd dd ...: Durchsuchen des Speichers ab Adresse $aaaaaa nach einer

Byte-Folge dd dd ... Die Suche wird durch einen beliebigen Tastendruck abgebrochen. Um das Auffinden von ASCII-Strings zu erleichtern, besteht hier die Möglichkeit der direkten Zeichenfolgeeingabe nach oben beschriebenem Muster. Zum Beispiel: Q aaaaaa „hallo" sucht das Wort „hallo" ab der Adresse $aaaaaa im Speicher. Die Mischung aus Bytes in hexadezimaler Darstellung und ASCII-Daten kann beliebig und bis zu 128 Zeichen lang sein, wobei als Trennzeichen zwischen den Einzelteilen der Bytefolge jeweils ein oder mehrere Abstände sein müssen.

R aaaaaa: Die Datei, die mit dem „F"-Befehl ausgewählt wurde, wird von der Adresse $aaaaaa ab in den Speicher des mc-68-ECB geladen. Dabei kann $aaaaaa beliebig innerhalb des 68000-Adreßbereichs liegen, so daß der Benutzer selbst auf eventuelle Kollisionen der zu ladenden Datei mit dem Stack des Monitors, dem RAM-Floppybereich oder dem Z80-Adreßraum zu achten hat.

S aaaaaa: Nach Eingabe dieses Befehls wird die Adresse $aaaaaa und danach das an dieser Stelle im Speicher befindliche Byte angezeigt. Anschließend erwartet der Monitor eines der folgenden Kommandos:
- Eine leere Eingabe (nur ⟨CR⟩) schreitet ohne Speicheränderung zur nächsten Adresse weiter.
- Ein Minuszeichen führt ohne Speicheränderung einen Adreß-Schritt zurück.
- Ein Punkt (alleine in einer Zeile) beendet das Kommando.
- Ein Datenbyte oder eine Folge von Datenbytes wird in den Speicher eingetragen. Die Möglichkeit eine ASCII-Datei einzugeben besteht ebenfalls.

T: Ausführen eines Einzelschrittes im Anwenderprogramm ab dem aktuellen PC. Anschließend werden die Inhalte sämtlicher CPU-Register auf dem Terminal angezeigt.

Tn: Tracen eines Programms ab dem aktuellem PC n-Befehle weit, dann Anzeige der Registerinhalte. n kann maximal $FFFF sein.

W aaaaaa bbbbbb: Schreiben eines Speicherbereichs von Adresse $aaaaaa bis $bbbbbb in eine Datei. Der Dateiname und eventuell das Laufwerk muß vorher durch das „F"-Kommando spezifiziert worden sein. Die Länge des abgespeicherten Bereichs wird immer auf volle Sektoren aufgerundet.

X: Anzeige sämtlicher CPU-Registerinhalte.

X Reg.: Anzeige des Registers Reg. (D0–A7, SP, ST, PC). Anschließend kann ein Datum in das entsprechende Register eingegeben werden. Eine leere Eingabe läßt das Register unverändert.

Z aaaaaa bbbbbb dd: Füllen des Speichers von $aaaaaa bis $bbbbbb mit dem Byte dd.

Z aaaaaa bbbbbb wwww: Speicher von $aaaaaa bis $bbbbbb mit Wort wwww füllen.

Z aaaaaa bbbbbb llllllll: Speicher mit Langwort llllllll füllen.

Die Unterprogramme des Monitors

Der Monitor befindet sich an der absoluten Adresse $200200 im EPROM auf der mc-68-ECB-Platine. Ganz am Anfang ist eine Sprungtabelle auf die wichtigsten Unterprogramme, die von Anwenderprogrammen ebenfalls einfach verwendet werden können, eingerichtet. Im folgenden werden kurz die Einsprungadressen, die Funktionen und die notwendigen Parameter dieser Unterprogramme besprochen. Alle CPU-Register bleiben (bis auf die zur Parameterübergabe notwendigen) unverändert.

$200200: Kaltstartinitialisierung des Monitors. Hierbei werden sämtliche Interruptvektoren in der Base-Page des 68000 und der gesamte RAM-Bereich des Monitors neu initialisiert.

$200202: Monitorwarmstart. Es wird lediglich der Stackpointer auf den Monitorstackbereich gerichtet. Danach meldet sich der Monitor mit seinem Prompt und erwartet eine Kommandoeingabe. Anwenderprogramme, die eine eigene Stackverwaltung haben, springen nach ihrem Abschluß am besten diese Routine an, um in den Monitor zurückzukehren.

$200204: Zeicheneingabe. Mit diesem Unterprogramm können Zeichen ohne Echo von der Konsole eingelesen werden. Die Zeichen werden im Register D0.B an das Benutzerprogramm übergeben.

$200208: Zeichenketteneingabe. Dieses Unterprogramm ermöglicht die Eingabe einer vollständigen Zeichenkette, die mit einem ⟨CR⟩ abgeschlossen werden muß. Durch ⟨BS⟩ (Backspace) können Fehleingaben rückgängig gemacht werden. Ob die eingegebenen Zeichen auch an das Terminal geschickt werden, hängt vom Inhalt des Systemflags mit der Adresse $48A ab. Steht hier ein $0000, so erfolgt ein Echo auf der Konsole, andernfalls wird es unterdrückt. Als notwendiger Eingabeparameter muß im Register A0 die Eingabepufferadresse übergeben werden. Der eingelesene String wird mit $0 abgeschlossen. Es wird kein CPU-Register verändert, so daß A0 bei der Rückkehr aus dem Unterprogramm auf den Anfang der eingelesenen Zeichenkette zeigt.

$20020C: Zeichenausgabe. Unterprogramm zur Zeichenausgabe auf das Terminal. Wurde vor dem Aufruf der Routine der Drucker aktiviert (durch „P"-Kommando) so erfolgt die Ausgabe auch auf den Drucker. Das Zeichen wird im Register D0.B übergeben.

$20020E: Ausgabe einer Zeichenkette. Völlig analog zur Eingabe einer Kette wird mit dieser Routine eine Zeichenkette an das Terminal ausgegeben. A0 zeigt auf den Beginn der auszugebenden Zeichenkette, die mit einem $0 abgeschlossen werden muß. Nach der Rückkehr aus dem Unterprogramm zeigt A0 hinter das Abschlußzeichen der Kette. Für den Drucker gilt dasselbe wie bei der Ausgabe eines einzelnen Zeichens.

$200210: Ausgabe eines ⟨CR-LF⟩ an das Terminal.

$200212: Hex-Zahl aus einem ASCII-String lesen. Diese Routine liest eine Hexzahl mit maximal 8 Ziffern aus einem ASCII-String. Das Register A0 dient als Zeiger (die Hexzahlen im String dürfen nicht, wie es hier im Text zur Verdeutlichung gemacht wird, mit einem $-Zeichen beginnen!) Vor dem Zahlenstring befindliche Leerzeichen ($20) werden überlesen. Als Abschlußzeichen für die Hexzahl gelten ein Leerzeichen ($20) oder das Zeichenkettenende-Zeichen ($0). Das Ergebnis der Wandlung wird im Register D0 zurückgeliefert. Register D7.B enthält die Zahl der eingelesenen Stellen, so daß leicht entschieden werden kann, ob ein Byte, ein Wort oder ein Langwort eingelesen wurde. Die Richtigkeit der Eingaben wird durch die Flags „Zero" und „Overflow" angezeigt, wobei folgende Zuordnung gilt: Z = 1, gültige Hexzahl; Z = 0, V = 0, nur ⟨CR⟩ eingegeben; Z = 0, V = 1, ungültiges Zeichen. Das Register A0 zeigt nach der Wandlung einer gültigen Hexzahl auf das Zeichen nach der Ziffernkette. Ent-

hält die ASCII-Darstellung der Zahl ein ungültiges Zeichen, so zeigt A0 nach der Rückkehr aus der Routine auf dieses Zeichen und durch die Flags wird der Fehler gemeldet. Enthält der String mehr als 8 Digits, so werden die ersten 8 Digits gewandelt.

Der für den Monitor notwendige Schreib-Lese-Speicher befindet sich zwischen Adresse $400 und ungefähr $700. In diesem Bereich werden alle Monitorvariablen gespeichert und Puffer für Ein- und Ausgaben aufgebaut. Der restliche Schreib-Lese-Speicher, soweit er nicht von der RAM-Floppy genutzt wird, steht für Anwenderprogramme zur Verfügung. Die Gliederung des Monitor-RAMs kann aus dem Auszug des Monitorassemblerlistings (*Bild 1*) entnommen werden. Da die absoluten Adressen erst durch den Linker erzeugt werden, muß man zu allen Adreßangaben $400 addieren.

Für den Anwender sind folgende Adressen wichtig:

errorlab: Hier steht der absolute Sprungbefehl zur Fehlerbehandlungsroutine bei falschen Eingaben. Wenn man als Anwender die vorhandenen Monitorkommandos durch eigene erweitern will, so braucht man nur diesen Sprung auf eine eigene Kommando-Decoder-Routine richten, die dann entscheiden muß ob ein neu implementierter Befehl vorliegt, oder wirklich eine Fehlermeldung ausgegeben werden soll, weil das vorgefundene Zeichen keine Abkürzung für einen zusätzlichen Befehl darstellt. Das gelesene ASCII-Kommandokürzel steht im Register D0.B.

sysflg: Durch Verändern des Inhalts dieses Wortes kann der Systemdrucker den Terminalausgaben parallel geschaltet werden. Es gilt die Zuordnung: sysflg = 0: Drucker aus; sysflg ⟨⟩ 0: Drucker ein.

outvec: Vektor auf die Monitorzeichenausgabe-Routine. Durch Verbiegen dieses Vektors auf eine eigene Routine können sämtliche Monitorausgaben beliebig umgelenkt werden. Die Zeichen werden wie üblich im Register D0.B angeliefert.

invec: Völlig analog zur Zeichenausgabe kann hier die Zeicheneingabe durch den Anwender seinen Wünschen entsprechend modifiziert werden. Die Zeichen müssen im Register D0.B geliefert werden. Sowohl bei der Ein- wie auch bei der Ausgabe dürfen die Inhalte der anderen CPU-Register nicht verändert werden.

inbuf: An dieser Adresse beginnt der Monitoreinlese-Puffer, in dem sämtliche Benutzereingaben bis zur Interpretation zwischengespeichert werden. Hier können z. B. bei der Einrichtung von eigenen Monitorbefehlen die zusätzlichen Parameter abgeholt werden, die in der Kommandozeile mit eingegeben wurden.

outbuf: Dieser Puffer nimmt alle auszugebenden ASCII-Zeichen auf, bevor sie durch die Zeichenkettenausgabe-Routine ans Terminal gesandt werden. Er kann auch von Anwenderprogrammen als Zwischenspeicher für Zeichenausgaben verwendet werden.

```
                        .even
                    *       Systemspeicher
00000000        errorlab:       .ds.b   6
00000006        dirbuf:         .ds.b   128

00000086        fcbflag         .ds.w   1
00000088        prflag:         .ds.w   1
0000008A        sysflg:         .ds.w   1       * bit 8 = 1 kein Eingabeecho
0000008C        trcnt:          .ds.w   1       * Zähler für Traceschritte
0000008E        disadr:         .ds.l   1       * Speicher für Adressen
00000092        distadr:        .ds.l   1
00000096        dissadr:        .ds.l   1
0000009A        regs:           .ds.l   15      * Puffer für d0 - a6
000000D6        sspr:           .ds.l   1       * stack pointer (a7=ssp)
000000DA        uspr:           .ds.l   1
000000DE                        .ds.w   1
000000E0        srr:            .ds.w   1       * Statusreg. nur 16 bit
000000E2        pcr:            .ds.l   1       * program counter
000000E6        brkp:           .ds.w   1
000000E8                        .ds.l   brknr   * breakpoint Adressen
00000110        brkp1:          .ds.l   brknr   * und gespeicherter Opcode
00000138        outvec:         .ds.l   1       * Vektoren für Ascii Ein und
                                                * Ausgabe
0000013C        invec:          .ds.l   1
00000140        inbuf:          .ds.b   buflen+1        * i/o Puffer
00000192        .even
00000192
00000192        outbuf:         .ds.b   buflen+1
000001E4        .even
000001E4
000001E4        stackspace:     .ds.l 50
000002AC        monst:          .ds.l   1       * Stackbereich

                .end
```

Bild 1. Diese Parameter benötigt der Monitor im mc-68-ECB

CP/M-2.2 und mc-68-ECB

Prinzipiell erfolgt der Steuerungsablauf zwischen den Prozessoren im Monitor-Betrieb ganz ähnlich wie bei der RAM-Floppy. Diesmal soll jedoch nicht der 68000 der Knecht der kleinen CPU sein, sondern umgekehrt die Geschicke des Gesamtsystems steuern und der Z80 die niedrigen Dienste an der Peripherie leisten. Zu diesem Zweck mußte eine leistungsfähige Schnittstelle geschaffen werden, die die Kommunikation zwischen den CPUs auch bei komplexeren Aufgaben übersichtlich gestaltet. Die Software sollte sowohl die BDOS- als auch die BIOS-Funktionen direkt verwenden. Außerdem sollte die Parameterübergabe für die BDOS-Funktionen, solange alle Daten direkt in den CPU-Registern gehalten werden können, völlig analog zum CP/M 68k laufen. Hier macht sich die direkte Ableitung des CP/M 68k aus dem 8-Bit-CP/M angenehm bemerkbar. Die Nummern der BDOS-Funktionen und die Zahl der Parameter beim Aufruf der Funktionen sind unter beiden Betriebssystemen identisch (soweit es die jeweiligen Funktionen unter CP/M 2.2 gibt), so daß nur die parametertragenden Register von einer CPU in die entsprechenden Register der anderen CPU kopiert werden müssen. Durch diesen Aufbau der Schnittstelle wird die später noch beschriebene Installation des Betriebssystems 68k zum Kinderspiel.

Zum Aufruf aller BDOS-Routinen von CP/M 2.2 müssen ins Register D0.B des

68000 die Funktionsnummer und ins Register D1 die jeweiligen Daten (falls vorhanden) geladen werden. Anschließend kann einfach durch einen Softwareinterrupt (Trap $4, Opcode $4E44) die jeweilige Routine ausgeführt werden. Ergebnisse, soweit sie in einem CPU-Register zu halten sind, werden in Register D0 geliefert. Zur Übertragung der 128-Byte-Sektoren von der Diskette oder von Ein-/Ausgabezeichenketten müssen die Informationen über einen Zwischenspeicher im Z80-Raum transferiert werden. Die rechtzeitige Bereitstellung bzw. Abholung der Daten vor bzw. nach Funktionsaufrufen obliegt dem Anwenderprogramm. Die Stelle im Speicher, an der die Datenübergabe stattfindet ist völlig beliebig, da für Diskettenzugriffe die DMA-Adresse und für die Ein-/Ausgabe von Zeichenketten die Pufferadressen dem BDOS vorher mitgeteilt werden müssen. Das Anwenderprogramm muß dann nur noch den Datenaustausch zwischen Z80- und 68000-Speicher gewährleisten. Die genaue Zuordnung der Register zu den Speicherplätzen beim Aufruf und umgekehrt bei der Rückkehr aus dem BDOS-Unterprogramm ist in *Tabelle 2* dargestellt. Die absoluten Adressen können aus dem Auszug aus dem Z80-Monitortreiber entnommen werden (*Tabelle 3*).

Tabelle 3: Vereinbarungen im Z80-Monitor-Treiber

```
                (0100)  0024                    org     100h
                        0025
0100            C3BB01  0026                    jp      start
                        0027
0103            00      0028    combyte         db      0
0104            00      0029    ready           db      0
0105            00      0030    data0           db      0       ; Parameterübergabefeld d1
                                                                ; bit 8–15
0106            00      0031    data1           db      0       ; d1 bit 0–7
0107            00      0032    data2           db      0       ; d2 bit 8–15
0108            00      0033    data3           db      0       ; d2 bit 0–7
0109            00      0034    data4           db      0       ; d3 bit 0–7
010A            (0080)  0035    dmasec          ds      128     ; Sektor Puffer
018A            (0024)  0036    fcb             ds      36      ; File Control Block für BDOS
01AE            0000    0037    sdsknr          dw      0
01B0            0000    0038    strkadr         dw      0
01B2            0000    0039    ssecadr         dw      0
01B4            0000    0040    sdmaadr         dw      0
01B6            0000    0041    ssectran        dw      0
                        0042
01B8            20      0043    floc            db      20h
01B9            02      0044                    db      02h
01BA            00      0045                    db      00h     ; Einsprungadr. 68000 Monitor
```

Tabelle 2: Funktionsaufrufe

Aufruf einer BDOS-Funktion durch den 68000		
68000-Register	→	Z80-Speicher
D0 (0... 7)	→	combyte
D1 (0... 7)	→	data1
D1 (8...15)	→	data0
D2 (0... 7)	→	data3
D2 (8...15)	→	data2
D3 (0... 7)	→	data4
Z80-Speicher	→	Z80-Register
data0	→	D
data1	→	E
combyte	→	C
Rückkehr nach dem BDOS-Aufruf		
Z80-Register	→	Z80-Speicher
L	→	data2
H	→	data3
B	→	data0
A	→	data1
Z80-Speicher	→	68000-Register
data1	→	D0 (0... 7)
data0	→	D0 (8...15)
Bei BDOS-Aufrufen werden folgende Register von einer CPU in die andere kopiert:		
68000-Register	→	Z80-Register
D0 (0... 7)	→	C
D1 (0... 7)	→	E
D1 (8...15)	→	D
Bei der Rückkehr gilt die Zuordnung:		
Z80-Register	→	68000-Register
A	→	D0 (0... 7)
B	→	D0 (8...15)

Wie man sieht, werden bei BDOS-Aufrufen nicht alle in den Z80-Speicher übertragenen Register verwendet, so daß dem Anwender noch Möglichkeiten für selbstgestrickte Parameterübergaben offenstehen. Diese, aus den 68000-Registern D2 und D3 zusätzlich übertragenen Informationen werden erst später bei direkten Zugriffen des CP/M 68k auf das Z80-BIOS ausgenutzt. Es soll aber bereits jetzt festgestellt werden, daß BIOS-genauso einfach wie BDOS-Zugriffe sind. Die BIOS-Funktionen können über den Softwareinterrupt Trap $5 (Opcode $4E45) aufgerufen werden. Dabei verläuft die Parameterübergabe zwischen 68000-Registern und Z80-Speicher genauso wie bei BDOS-Zugriffen. Lediglich die Bedeutung der in data0 bis data4 übertragenen Informationen hängt von der angesteuerten BIOS-Funktion ab.

Diese Zusammenhänge sollen aber erst mit der Installation des Betriebssystems besprochen werden. Die gesamte Aufteilung des Z80-Adreßraums, in dem die Parameter- und Datenübergabe zwischen den CPUs stattfindet, ist in Tabelle 3 dargestellt, und darf nicht verändert werden, da der Monitor von festen Adressen ausgeht.

Wie jedes CP/M-Programm beginnt auch dieses bei $ 100 mit einem Sprung auf den Programmanfang. Im Anschluß be-

```
                0002 ;      Biostreiber und Start für SWS 68000
                0003 ;
                0004 ;      Version 2.0     11.6.86  mit Bdos-Zugriff
                0006
      (0005)    0007 ntry   equ    5        ; BDOS Eingang
                0008
                0009 ;      Biosfunktionsnummern
                0010
      (000E)    0011 wrt    equ    14
      (000D)    0012 red    equ    13
      (0010)    0013 stran  equ    16
      (00FF)    0014 quit   equ    0ffh     ; Endekommando vom 68000 aus
                0015
      (0000)    0016 clr    equ    0
      (00FF)    0017 readyz80       equ    0ffh
      (00FF)    0018 zlatch equ    0ffh
                0019
      (0000)    0020 wboot  equ    0
                0021
                0022
      (0004)    0023 begin  equ    4              ; Go Kommando an den 68000 Ramfloppytreiber
      (0010)    0024 parfield       equ    10h    ; Parameterfeldadresse für Ramfloppytreiber
                0025
      (0100)    0026        org    100h
                0027
0100  C3BB01    0028        jp     start
                0029
0103  00        0030 combyte db     0      ; übergabefeld für Funktionsnummern
0104  00        0031 ready  db     0      ; Funktion ausgeführt ?
0105  00        0032 data0  db     0      ; Parameteruebergabefeld d1 bit 8-15
0106  00        0033 data1  db     0      ; d1 bit 0-7
0107  00        0034 data2  db     0      ; d2 bit 8-15
0108  00        0035 data3  db     0      ; d2 bit 0-7
0109  00        0036 data4  db     0      ; d3 bit 0-7
010A  (0080)    0037 dmasec ds     128    ; Sektor Puffer
018A  (0024)    0038 fcb    ds     36     ; file control block (FCB)
01AE  0000      0039 sdsknr dw     0
01B0  0000      0040 strkadr dw    0
01B2  0000      0041 ssecadr dw    0
01B4  0000      0042 sdmaadr dw    0
01B6  0000      0043 ssectran       dw     0
                0044
01B8  20        0045 _floc  db     20h
01B9  02        0046        db     02h
01BA  00        0047        db     00h    ; Einsprungadesse in Monitorprogramm
                0048
01BB  010300    0049 start: ld     bc,3           ; Kopiere Einsprungadresse in Paramterfeld
01BE  21B801    0050        ld     hl,_floc       ; für GO-Kommando
01C1  111100    0051        ld     de,parfield+1
01C4  EDB0      0052        ldir
01C6  AF        0053        xor    a
01C7  321000    0054        ld     (parfield),a   ; Ready-Byte im Parameterfeld löschen
01CA  3E04      0055        ld     a,begin        ; Start 68000
01CC  D3FF      0056        out    zlatch,a
01CE  3A1000    0057 start1: ld    a,(parfield)   ; Warten auf Kommandoausführung
01D1  B7        0058        or     a
01D2  28FA      0059        jr     z,start1
01D4  3E09      0060        ld     a,9            ; Bios Funktionsnummer für 'seldsk'
01D6  CD4C02    0061        call   getadr
01D9  22AE01    0062        ld     (sdsknr),hl    ; Direkte Einsprungadresse in BIOS
01DC  23        0063        inc    hl             ; Sprungleiste abspeichern
01DD  23        0064        inc    hl
01DE  23        0065        inc    hl
01DF  22B001    0066        ld     (strkadr),hl   ; 'settrk'
01E2  23        0067        inc    hl
01E3  23        0068        inc    hl
01E4  23        0069        inc    hl
01E5  22B201    0070        ld     (ssecadr),hl   ; 'setsec'
01E8  23        0071        inc    hl
01E9  23        0072        inc    hl
01EA  23        0073        inc    hl
01EB  22B401    0074        ld     (sdmaadr),hl   ; 'setdma'
01EE  3E10      0075        ld     a,16
01F0  CD4C02    0076        call   getadr         ; Adresse für Funktion 'sectran' berechnen
01F3  22B601    0077        ld     (ssectran),hl  ; 'sectran'
                0078
                0079
01F6  210401    0080 comint: ld    hl,ready       ; Anfang des Z80 Kommandointerpreters
01F9  7E        0081 waitlp: ld    a,(hl)         ; Warten auf ein Kommando
01FA  B7        0082        or     a
01FB  28FC      0083        jr     z,waitlp       ; Offh --> Kommando vom 68000
01FD  3A0301    0084        ld     a,(combyte)    ; Funktionsnnummer holen
0200  FEFF      0085        cp     quit           ; Vergleich mit <Ende> Befehl
0202  CA0000    0086        jp     z,wboot        ; ja --> Zurück ins 8 Bit CP/M
0205  CB7F      0087        bit    7,a            ; Bit 7 gesetzt --> BDOS Anforderung
0207  2034      0088        jr     nz,bdosent
0209  CD4C02    0089        call   getadr         ; Adresse der BIOS Funktion berechnen
020C  E5        0090        push   hl             ; und auf Stack speichern
020D  210301    0091        ld     hl,combyte
0210  3E0E      0092        ld     a,wrt          ; Schreibbefehl ?
0212  BE        0093        cp     (hl)
0213  2847      0094        jr     z,write
0215  3E0D      0095        ld     a,red
0217  BE        0096        cp     (hl)           ; Lesebefehl ?
0218  2842      0097        jr     z,read         ; Schreiben und Lesen erfordert besondere
                0098                              ; Behandlung
021A  3E10      0099        ld     a,stran
021C  BE        0100        cp     (hl)
021D  286A      0101        jr     z,sectran      ; Ebenso 'sectran_
021F  210601    0102        ld     hl,data1       ; hole D1 (0 - 7)
0222  4E        0103        ld     c,(hl)         ; Eingabeparameter für cpm22 bios
0223  E1        0104 entry: pop    hl             ; Hole Einsprungadresse
0224  CD5B02    0105        call   callhl         ; Funktion ausführen
0227  5C        0106 entr1: ld     e,h            ; Ergebnisse aus BDOS und BIOS Aufrufen
0228  65        0107        ld     h,l            ; in Parameterübergabefeld schreiben
0229  6B        0108        ld     l,e
022A  220701    0109        ld     (data2),hl
                0110
                0111 ;      16 Bit BIOS oder BDOS Antwort
                0112
022D  68        0113        ld     l,b            ; High Byte BDOS Antwort
022E  67        0114        ld     h,a            ; Low Byte BDOS Antwort
022F  220501    0115        ld     (data0),hl     ; werden in Data0 geladen
0232  210401    0116 exit:  ld     hl,ready       ; ready-Byte zurücksetzen
0235  3600      0117        ld     (hl),clr       ;
0237  3EFF      0118        ld     a,readyz80     ; Signal an 68000 . Alles fertig !
0239  D3FF      0119        out    zlatch,a
023B  18B9      0120        jr     comint         ; Sprung an den Anfang des Kommandointerpreters
                0121
                0122
                0123 bdosent:
023D  CBBF      0124        res    7,a            ; Bit 7 in der BDOS Funktionnummer rücksetzen
023F  4F        0125        ld     c,a            ; Kommandonummer in Register C
0240  ED5B0501  0126        ld     de,(data0)     ; Eingabeparameter laden
```

Bild 2. BIOS-Treiber und Start für den mc-68-ECB

finden sich gleich die notwendigen Speicherplätze für den Parameter- und Datenverkehr. Im „combyte" werden die jeweiligen Nummern für die BIOS- bzw. BDOS-Funktionen an den Z80 übergeben. Dabei ist zu beachten, daß die Funktionsnummern für das BIOS von 1 (Warmstart) bis 16 (Sectran) reichen, und für den Aufruf der BDOS-Funktionen Bit 7 im „combyte" gesetzt sein muß, d. h. zu den BDOS-Funktionsnummern muß $ 80 addiert werden. Dies geschieht automatisch bei der Verwendung des Softwareinterrupts Trap #4.

Das an Adresse $ 104 stehende „ready"-Byte dient dazu, dem Z80 das Vorliegen eines gültigen Kommandos anzuzeigen. Sobald hier eine Zahl ungleich 0 steht beginnt der Z80 mit der Kommandoausführung.

Die Bedeutung der Speicherplätze data0 bis data4 für den Datenaustausch wurde oben bereits ausführlich erläutert.

Im Anschluß daran folgt ein 128 Byte langer Speicherraum, über den bei Lese- oder Schreibbefehlen der Transfer der Daten stattfindet. Der Z80 ließt oder schreibt bei sämtlichen Diskettenzugriffen aus oder in diesen Speicherbereich. Hier kann der 68000 die Daten also ablegen oder abholen. Ab der Adresse $ 18A beginnt der 36 Bytes lange FCB, der die notwendigen Informationen bei Dateizugriffen enthält. Dieser FCB wird durch das Monitorkommando „F" aufgebaut. Die im Anschluß daran folgenden fünf Worte im Speicher enthalten nach der Initialisierung des Z80-Programms direkte Einsprungadressen in die BIOS-Sprungleiste für bestimmte Funktionen zur Steuerung der Laufwerke. Den Abschluß des Feldes bilden 3 Byte, die die Einsprungadresse in den 68000-Monitor enthalten. Dieser beginnt bei der 68000-Adresse $200200. Der vollständige Programmausdruck des Z80-Treibers für den 68000-Monitor ist in *Bild 2* abgedruckt. Durch die reichlichen Kommentare ist es sicherlich jedem möglich den genauen Steuerungsablauf auf der Z80-Seite zu verfolgen. Bei der Installation des 68k-Betriebssystems bleibt dieser Treiber bis auf die Einsprungadresse des 68000-Programms unverändert. Dieser Zeiger muß dann natürlich auf die Kaltstartroutine des Betriebssystems gerichtet werden.

```
0244  7B        0127          ld      a,e
0245  5A        0128          ld      e,d
0246  57        0129          ld      d,a             ; Austausch d - e
0247  CD0500    0130          call    ntry            ; Eingang CPM80 Bdos
024A  18DB      0131          jr      entr1           ;
                 0132
                 0133  getadr:                 ; Berechne Einsprungadressen für BIOS-Funktionen
024C  C5        0134          push    bc
024D  3D        0135          dec     a
024E  47        0136          ld      b,a
024F  80        0137          add     a,b
0250  80        0138          add     a,b
0251  010000    0139          ld      bc,0
0254  4F        0140          ld      c,a
0255  2A0100    0141          ld      hl,(wboot+1)    ; Hole Beginn der BIOS-Sprungleiste
0258  09        0142          add     hl,bc
0259  C1        0143          pop     bc
025A  C9        0144          ret
                 0145
025B  E9        0146  callhl: jp      (hl)            ; Simuliert Befehl 'call (hl)'
                 0147
                 0148
                 0149  read:
025C  3A0901    0150  write:  ld      a,(data4)       ; Hole Drive Nummer
025F  4F        0151          ld      c,a
0260  2AAE01    0152          ld      hl,(sdsknr)
0263  CD5B02    0153          call    callhl          ; Übergabe an BIOS
0266  210501    0154          ld      hl,data0        ; Hole Track
0269  46        0155          ld      b,(hl)
026A  23        0156          inc     hl
026B  4E        0157          ld      c,(hl)
026C  2AB001    0158          ld      hl,(strkadr)    ; Übergabe an BIOS
026F  CD5B02    0159          call    callhl
0272  210701    0160          ld      hl,data2
0275  46        0161          ld      b,(hl)          ; Hole Sektornummer
0276  23        0162          inc     hl
0277  4E        0163          ld      c,(hl)
0278  2AB201    0164          ld      hl,(ssecadr)    ; Übergabe an BIOS
027B  CD5B02    0165          call    callhl
027E  010A01    0166          ld      bc,dmasec
0281  2AB401    0167          ld      hl,(sdmaadr)
0284  CD5B02    0168          call    callhl          ; Setze DMA-Adresse auf Sektorpuffer am
0287  189A      0169          jr      entry           ; Programmanfang
                 0170
0289  ED4B0501  0171  sectran:       ld      bc,(data0)      ; Hole die logische Sektornummer
028D  78        0172          ld      a,b
028E  41        0173          ld      b,c
028F  4F        0174          ld      c,a
0290  ED5B0701  0175          ld      de,(data2)              ; Hole die Sektorversatztabellen-
0294  7A        0176          ld      a,d                     ; adresse
0295  53        0177          ld      d,e
0296  5F        0178          ld      e,a
0297  188A      0179          jr      entry                   ; Sprung ins BIOS zur Berechnung der
                 0180                                          ; physikalischen Sektornummer
0299  (0000)    0181          end

Errors          0
Range Count     0
```

Im Hexdump (*Bild 3* und *Bild 4*) des kompletten Monitorprogramms im 68000-System ist so dargestellt, wie er in zwei 8-Bit-EPROMs programmiert werden muß. Die angegebenen Adressen beziehen sich auf die relativen EPROM-Adressen. Am Ende jeder Zeile befindet sich eine 2-Byte-Prüfsumme, die das Auffinden von Fehlern erleichtert. Zur Kontrolle der gesamten Eingaben ist im Anschluß an den Dump die Prüfsumme über alle Byte ausgedruckt.

Im nächsten Teil dieser Artikelserie wird die Installation des CP/M 68k besprochen werden. Der größte Vorteil bei der Verwendung dieses Systems ist die vollständige Kompatibilität der Fileverwaltung auf den Disketten, so daß ein Wechsel zwischen den beiden Betriebssystemen (8-Bit- und 16-Bit-CP/M) durch einen einfachen Programmaufruf erreicht werden kann. Die 8- und 16-Bit-Files stehen völlig gemischt auf den Disketten. Dadurch kann man die 8-Bit-Software wie bisher weiterverwenden und zum Beispiel zum Erstellen einer 68000-Assemblerdatei die gewöhnten Editoren benützen. Dieser Sourcefile kann dann, nach dem Wechsel ins 16-Bit-System, direkt vom 68000-Assembler übersetzt werden.

Dieses Beispiel soll aber nur einen kleinen Einblick in die vielfältigen Möglichkeiten und Vorteile des Gesamtsystems geben. Wie es genau gemacht wird steht in der nächsten mc. ▶

```
000100  14 30 00 E6 00 80 56 36 66 00 F4 7C 00 AC 00 88  :.0.f..V6f.tö.,..:  40
000110  00 DE 46 FC 00 4C 7C 20 C7 12 7C 00 AC 3C 23 24  :.^Fü.Lö G.ö..,<#$:  8C
000120  00 20 EE E7 40 7C 00 38 51 50 18 06 91 C9 F8 DF  :. ng§ö.8QP...Ix_:  D9
000130  03 75 E7 80 7C 00 38 50 90 DF 00 75 E7 00 0D E4  :.ug.ö.8P._.ug..d:  9F
000140  0A E0 DF 01 75 E7 C0 7C 00 3C 51 50 91 3C 18 0A  :.`_.ug§ö.<QP.<..:  2E
000150  3C 50 F2 30 F6 3C 08 0A 3C 50 E2 20 DE 3C 0D 10  :<Pr0v<..<Pb ^<..:  B7
000160  C0 79 00 8A 02 98 C9 CA 10 DF 03 75 81 88 08 84  :§y....IJ._.u....:  EC
000170  20 80 08 00 7C 75 E7 80 7C 00 3C 50 90 DF 00 75  : ...üug.ö.<P._.u:  EC
000180  E7 C0 7C 00 00 7C 20 5C D9 D9 7C 00 86 FC FF 58  :g§ö..ö öYYö..ö.X:  22
000190  58 58 11 98 C8 FC FC 00 AC FC 00 AC 58 FC 00 98  :XX..Höö.,ö.,Xö..:  59
0001A0  58 09 98 98 C8 FA 7C 20 EC D9 D9 DF 01 75 F9 20  :X...Hzö ìYY_.uy :  FB
0001B0  F8 E7 80 7C 00 40 00 1C 06 DF 00 75 E7 60 7C 20  :ug.ö.§..._.ug`ö :  74
0001C0  F4 64 18 3C 60 04 3C 20 19 59 3C 00 04 00 F2 92  :td.<`.< .Y<...r.:  A2
0001D0  DF 03 75 81 3C 09 7C 00 08 7C 20 C4 D9 C9 FC FC  :_.u.<.ö..ö DYIöö:  9B
0001E0  20 AC 00 90 FC 20 B0 00 94 FC 20 26 00 98 FC 20  : ,..ö 0..ö &..ö :  B2
0001F0  A2 00 B8 75 00 18 3C 20 F8 88 1F 75 08 7C 20 FF  :".8u..< x..u.ö .:  FA
000200  00 44 5F 75 E7 00 81 D8 07 10 3E 3C 20 38 3C 30  :.D_ug..X..>< 8<0:  AD
000210  3E 3C 39 12 3C DF 3C 41 2E 3C 46 28 00 3C 0F 88  :><9.<_<A.<F(.<..:  08
000220  81 00 CA CC 01 07 42 FC 04 10 3C 07 EE FC 00 04  :..JL..Bö..<.nö..:  A2
000230  FC 02 DF 06 75 40 0A 40 06 40 0A 40 00 58 02 1F  :ö._.u§.§.§.§.X..:  EB
000240  00 00 02 1F 3C 0F 3C 30 3C 39 02 00 C0 75 E7 60  :....<.<0<9..§ug`:  CB
000250  7C 20 4E 18 3C DF 88 18 19 59 7C 00 1E 3C 24 10  :ö N.<_...Yö..<$.:  39
000260  3C 30 EA 3C 37 E4 00 04 3C DF 40 D8 92 DF 03 75  :<0j<7d..<_§X._.u:  CD
000270  E7 F0 00 92 7C 20 78 7C 00 9A 7C 00 92 13 4B 03  :gp..ö xö..ö...K.:  02
000280  03 D9 CB FC 1A 00 5E 04 00 7A 82 04 C9 E6 10 4B  :.YKö..^..z..If.K:  29
000290  00 24 00 52 02 D0 39 00 E0 4B 7C 20 C4 3C 04 49  :.$.R.P9.`Kö D<.I:  95
0002A0  02 3A 4A CA F6 49 01 07 02 2A 4A 01 05 02 20 4A  :.:JJvI...*J... J:  7F
0002B0  10 41 07 01 00 1E 10 4B 00 D4 00 02 DF 0F 75 3C  :.A.....K.T._.u<:  53
0002C0  04 DA C8 FC 4A 75 01 20 00 C1 C8 FC 1F 75 3C 30  :.ZHöJu. .AHö.u<0:  07
0002D0  14 3C 28 0E 0F 0A 10 06 11 02 12 E7 E0 00 BC 81  :.<(........g`.<.:  DE
0002E0  00 81 7C 00 92 7C 00 9A 7C 20 78 03 F2 00 8A CA  :..ö..ö..ö x.r..J:  02
0002F0  F8 31 00 00 82 02 9E 10 7C 00 92 00 4E 7C 00 40  :x1.....ö...Nö.§:  73
000300  00 88 00 02 12 80 00 7C 00 DE 50 00 60 04 00 DA  :.......ö.^P.`..Z:  04
000310  DF 06 75 E7 F0 7C 00 92 00 D6 18 40 C0 00 8E 00  :_.ugpö...V.§§...:  BB
000320  C8 0A 89 00 2A 80 08 07 79 00 8E 00 00 1E 4B 0F  :H...*...y.....K.:  93
000330  E7 40 09 00 0A 02 00 1E 19 00 0C 01 00 12 CA F2  :g§............Jr:  4E
000340  03 00 08 DF 04 FC 27 19 3C 20 08 3C 7F 02 02 2E  :..._.ö'.< .<....:  7B
000350  C0 CA EA FC 27 18 4B 00 96 00 C4 00 0C C9 A0 C9  :§Jjö'.K...D..I I:  92
000360  00 8E 00 B0 E7 F0 7C 00 92 00 34 00 9A 79 00 92  :...0gpö...4..y..:  FC
000370  02 40 00 92 4B 09 00 84 02 00 98 11 00 86 02 00  :.§..K..........:  DF
000380  8C 18 4B 00 3E 7C 00 40 01 00 76 00 F0 18 47 0F  :..K.>ö.§..v.p.G.:  BE
000390  01 87 80 00 88 CF F8 C6 05 E2 08 05 AA 89 A6 10  :.....OxF.b..*.&.:  FA
0003A0  3C 27 14 05 88 18 3C 27 C4 00 8E C0 F0 3C 2D 04  :<'....<'D..§p<-.:  EE
0003B0  89 80 3C 2E 0C 00 8C 00 74 00 84 00 00 C9 00 92  :..<.....t....I..:  5E
0003C0  DF 07 75 E7 70 7C 00 92 00 76 00 86 40 7C 00 06  :_.ugpö...v..§ö..:  7E
0003D0  00 66 16 47 0F 01 87 80 00 88 CF F8 C6 E4 12 18  :.f.G......OxFd..:  FD
0003E0  27 58 D8 18 27 D4 20 F4 7C 00 06 09 88 3C 00 49  :'XX.'T tö....<.I:  16
0003F0  08 42 49 41 0A 00 D8 30 3C 00 08 E8 CA EC 00 7A  :.BIA..X0<..hJJ.z:  8D
000400  7C 00 92 0A 83 00 66 18 7C 00 92 00 2E C4 00 DA  :ö.....f.ö....D.Z:  F3
000410  00 56 DF 0F 75 00 DC 08 C0 00 E2 02 40 B9 07 00  :.V_.u.ö.§.b.§9..:  41
000420  E0 7C 00 E6 19 10 42 59 52 24 BC 4E CA F4 79 00  :`ö.f..BYR$<NJty.:  BD
000430  DA 60 F9 FF 00 9A 39 00 E2 39 00 E0 78 00 7C 75  :Z`y...9.b9.`s.öu:  64
000440  00 86 08 C0 00 8C 08 10 79 00 8C F9 07 00 E0 BC  :.`..§....y..y..`<:  93
000450  00 56 75 F9 FF 00 9A 68 C8 00 DA DF 00 E0 DF 00  :.Vuy...hH.Z_.`_.:  05
000460  E2 CF 00 D6 7C 00 AC 00 A8 00 0C 79 00 8C 00 56  :bO.Vö...(..y...V:  BE
000470  00 7A E7 60 10 01 DF 01 58 00 84 7C 00 E6 00 0A  :.zg`._.X..ö.f...:  FA
000480  22 80 00 81 19 41 0A 5E 49 C1 80 49 41 7C 00 E6  :"....A.^IA.IAö.f:  5B
000490  81 4E 48 19 38 42 41 7C 00 92 00 42 01 41 00 42  :.NH.8BAö...B.A.B:  BF
0004A0  FC 3A 02 00 44 19 00 24 10 7C 00 92 00 EC CA D0  :ö:..D..$.ö...lJP:  5D
0004B0  10 7C 20 14 00 DC 04 00 88 DF 07 75 18 00 8C 7C  :.ö ..ö...._.u...ö:  A3
0004C0  00 E6 19 E6 42 40 E0 48 C0 E9 04 CA FA 79 00 E6  :.f.fB§`Hßi.Jzy.f:  5F
0004D0  D0 F9 FF 00 9A 68 C8 00 DA DF 00 E0 DF 00 E2 B9  :Py.Z.hH.Z_.`_.b9:  A5
0004E0  00 E2 CF 00 D6 7C 00 AC 7C 00 E6 19 0C 42 59 A9  :.bO.Vö.,ö.f..BY):  7A
0004F0  24 CA F8 00 90 7C 20 07 00 54 00 EA 00 3A E7 60  :$Jx..ö ..T.j.:g`:  D8
```

Bild 3. Der Monitor, ungerade Adressen

```
000500  00 06 2E 40 00 FE 26 89 22 40 00 F2 1A 8A C9 08  :...§.ß&."§.r..I.:  EA
000510  D9 80 FA 10 C0 C0 21 80 FA 04 00 C2 DF 01 75 E7  :Y.z.§§!.z..B_.ug:  80
000520  40 00 C4 06 00 AE 3C 40 00 B6 00 A2 89 00 9C 00  :§.D...<§.ö."....:  B1
000530  00 A6 00 92 07 02 0E 07 04 10 C0 89 FA 0E C0 89  :.&........§.z.§.:  04
000540  FA 06 C0 89 FA DF 03 75 F9 FF 00 9A 7C 20 29 0E  :z.§.z_.uy...ö ).:  FF
000550  F9 FF 00 9A 7C 20 37 FC 00 00 C4 00 8E 68 C8 00  :y...ö 7ö..D..hH.:  E3
000560  DA 2F 0A C0 00 E2 00 A4 EF 08 00 E0 7C 20 19 00  :Z/.§.b.§o..`ö ..:  E5
000570  66 2F 02 00 8A 00 8C 00 F0 00 40 E7 80 7C 20 48  :f/.....p.§g.ö H:  2B
000580  46 E7 80 7C 20 5C 3A E7 80 7C 20 6E 2E E7 80 7C  :Fg.ö ö:g.ö n.g.ö:  61
000590  20 7E 22 E7 80 7C 20 91 16 E7 80 7C 20 AC 0A E7  : ß"g.ö ..g.ö ..g:  0A
0005A0  80 7C 20 C0 FC 00 00 2A 00 F4 DF 00 F9 FF 00 9A  :.ö §ö..*.t_.y...:  67
0005B0  68 C8 00 DA 2F 02 C0 00 E2 EF 00 00 E0 00 F6 00  :hH.Z/.§.bo..`.v.:  A2
0005C0  F8 00 5C 00 AC E7 00 41 00 01 3C 02 44 79 00 88  :x.ö..g.A..<.Dy...  AC
0005D0  08 07 3C 05 44 DF 82 75 01 3C FF 3C 06 44 00 F6  :..<.D_.u.<.<.D.v:  22
0005E0  1F 75 01 3C FF 3C 06 44 00 04 3C 01 1F 00 75 79  :.u.<.<.D..<...uy:  A4
0005F0  00 00 88 0C 7C 20 00 00 56 75 7C 20 EC F2 7C 50  :....ö ..Vuö ìröP:  41
000600  07 BC FF 7C FF 02 79 20 04 D1 FC FF 00 86 3C 19  :.<.ö..y .Qö...<.:  83
000610  44 00 00 C2 41 28 3A 01 18 10 48 01 FF 38 39 00  :D..BA(:...H..89.:  8B
000620  00 30 41 48 08 39 00 00 22 7C 00 8A 00 FC 3E 40  :.0AH.9.."ö...ö>§:  9C
000630  0A 7C 20 2D 00 DC 79 00 86 75 7C 20 46 00 CA 01  :.ö -.öy..uö F.J.:  D0
000640  61 04 01 20 01 0F 08 01 00 02 75 3C FF 75 E7 40  :a.. ......u<.ug§:  ED
000650  81 49 09 3C 22 11 49 CA FA 43 3C 0A BC 20 49 CA  :.I.<".IJzC<.< IJ:  C7
000660  F8 00 24 43 3C 08 18 1A 02 2E 14 02 60 04 02 20  :x.$C<.......`.. :  A1
000670  82 49 C9 E6 2C 02 2C 43 FC 00 10 3C 03 18 14 02  :.IIf,.,Cü..<....:  90
000680  60 04 02 20 82 49 C9 EC 04 40 04 3C FF DF 0E 75  :`.. .III.§.<._.u:  EB
000690  00 E6 00 D2 79 00 86 00 80 7C 00 8A 00 1C 29 18  :.f.Ry....ö....).:  9A
0006A0  29 40 40 3C 0F 3C 8A 44 3C FF 64 3C 1A 3C 0A 44  :)§§<.<.D<.d<.<.D:  7D
0006B0  3C 8A 7C 00 0A 49 00 E8 49 08 3C 14 44 00 08 00  :<.ö..I.hI.<.D...:  6A
0006C0  E0 42 EE 7C 20 99 00 B8 7C 00 92 09 40 00 D6 10  :`Bnö ..8ö...§.V.:  3A
0006D0  7C 00 92 00 9E 00 CC 75 7C 20 68 00 8E 7C 20 84  :ö.....Luö h..ä .:  9F
0006E0  00 84 00 42 00 2E 40 00 38 00 24 8A 00 1E 00 3C  :...B..§.8.$....<:  74
0006F0  8A 3C 13 44 3C 16 44 00 FF 40 3C 00 0A 3C 1A 44  :.<.D<.D..§<..<.D:  D2
000700  7C 00 8A 4E 29 18 29 40 41 42 3C 8A 09 2C 3C 15  :ö..N).)§AB<..,<.:  CD
000710  44 00 18 42 8A EE 3C 10 44 75 7C 20 A2 00 0A 7C  :D..B.n<.Duö "..ö:  DF
000720  20 3E 00 00 3C 7F 9A 49 CB FA 75 C9 FC 50 01 75  : >..<..IKzuIöP.u:  C1
000730  3C 7F D0 88 CB FA 75 C0 50 07 7C 50 0B 41 02 49  :<.P.KzuêP.öP.A.I:  C7
000740  81 42 06 4A 42 04 43 08 FC FF 50 09 39 FF 60 01  :.B.JB.C.ö.P.9.`.:  91
000750  F6 79 20 08 11 75 40 80 E7 C0 B8 10 48 28 02 DF  :vy ..u§.g§8.H(._:  9D
000760  0E 73 20 90 20 A0 20 F6 20 02 20 0E 20 1A 20 26  :.s .  v . . . &:  D7
000770  20 A6 20 32 20 3E 20 8A 20 B0 58 20 9C 44 20 26  : & 2 > . 0X .D &:  8E
000780  51 20 86 53 20 C8 47 20 2A 54 20 80 42 20 E4 4D  :Q .S HG *T .B dM:  4A
000790  20 FC 5A 20 3E 45 20 FC 50 20 DE 46 20 14 52 20  : öZ >E öP ^F .R :  6F
0007A0  20 57 20 C4 00 00 00 24 20 9C 24 20 A2 53 20 A8  : W D...$ .$ "S (:  3C
0007B0  53 20 AC 52 20 B0 43 20 B4 00 20 E0 30 20 31 20  :S ,R 0C 4. `0 1 :  F9
0007C0  32 20 33 20 34 20 35 20 36 20 37 20 30 20 31 20  :2 3 4 5 6 7 0 1 :  9C
0007D0  32 20 33 20 34 20 35 20 36 20 53 20 53 20 52 20  :2 3 4 5 6 S S R :  FC
0007E0  43 20 52 20 4F 4C 20 45 20 4E 47 20 58 20 54 41  :C R OL E NG X TA:  B7
0007F0  20 55 20 49 20 20 0A 72 6E 65 20 6E 20 21 0D 00  : U I  .rne n !..:  49
000800  0A 72 6E 65 20 66 20 21 0D 00 0A 6F 62 65 6B 6F  :.rne f !...obeko:  3D
000810  6E 73 73 74 21 0A 0D 49 72 67 6C 72 66 6C 6E 6D  :nsst!..IrglrflnM:  AD
000820  20 0D 00 0A 70 63 66 65 20 72 76 20 6F 20 6C 6F  : ...pcfe rv o lo:  67
000830  65 20 0D 00 0A 6F 66 6C 6E 6D 20 70 63 66 65 20  :e ...oflnm pcfe :  96
000840  0D 00 0A 69 65 6E 74 66 75 64 21 0A 0D 4D 78 3D  :...ientfud!..Mx=:  40
000850  00 0A 69 65 74 72 20 75 6C 21 0A 0D 44 73 20 75  :..ietr ul!..Ds u:  43
000860  6C 21 0A 0D 2A 2A 2A 20 43 36 2D 43 20 6F 69 6F  :l!..*** C6-C oio:  92
000870  20 65 73 3A 31 33 20 31 2E 2E 36 2A 2A 2A 0A 20  : es:13 1..6***. :  21
000880  3F 20 0A 62 65 6B 6F 6E 20 69 20 0A 61 63 73 20  :? .bekon i .acs :  82
000890  64 65 73 20 62 73 65 72 72 61 20 61 72 73 20 72  :des bserra ars r:  D3
0008A0  6F 20 74 00 6C 65 61 20 70 63 64 20 74 00 69 69  :o t.lea pcd t.ii:  F2
0008B0  69 6E 62 20 20 74 00 68 63 20 72 6F 20 74 00 76  :inb  t.hc ro t.v:  C3
0008C0  72 6C 77 65 72 72 61 20 70 69 69 65 65 20 6E 74  :rlwerra piiee nt:  2D
0008D0  75 74 6F 20 74 00 69 20 30 30 6F 63 64 20 74 00  :uto t.i 00ocd t.:  9F
0008E0  69 20 31 31 6F 63 64 20 74 00 00 00 00 00 00 00  :i 11ocd t.......:  B5

Prüfsumme insgesamt : 02A156
```

```
000100  60 60 60 00 60 00 60 60 60 60 01 2E 00 06 61 01 :'''.'.''''....a.: 97
000110  61 00 4E 46 20 61 20 00 10 61 2E 00 06 61 70 61 :a.NF a ..a...apa: 6D
000120  61 01 60 48 C0 22 00 05 22 72 10 67 4E 51 FF 4C :a.'H§".."r.gNQ.L: E6
000130  02 4E 48 00 20 00 05 20 4E 4C 01 4E 48 80 70 61 :.NH. .. NL.NH.pa: 5F
000140  70 61 4C 00 4E 48 C0 22 00 05 22 72 4E B0 00 66 :paL.NH§".."rNO.f: 92
000150  B2 00 67 61 60 B0 00 66 B2 00 67 61 60 B0 00 67 :2.ga'O.f2.ga'O.g: E1
000160  10 4A 00 04 66 61 51 FF 42 4C 03 4E 52 53 70 61 :.J..faQ.BL.NRSpa: CA
000170  70 61 70 61 FF 4E 48 00 20 00 05 20 4E 4C 01 4E :papa.NH. .. NL.N: 65
000180  48 80 20 00 04 22 00 03 20 30 20 00 04 30 00 42 :H. .."..O ..O.B: F7
000190  42 42 70 42 51 FF 20 00 06 20 00 06 42 30 20 42 :BBpBQ. .. ..BO B: A6
0001A0  42 70 42 42 51 FF 22 00 0E 20 20 4C 03 4E 4E 00 :BpBBQ."..  L.NN.: E1
0001B0  03 48 00 20 00 05 61 FF 61 4C 01 4E 48 C0 22 00 :.H. ..a.aL.NH§".: F6
0001C0  0E 61 10 B0 00 6D 90 00 32 24 B2 00 67 B2 66 4E :.a.O.m..2$2.g2fN: 01
0001D0  4C 06 4E 42 32 00 20 00 00 22 00 0E 20 51 FF 23 :L.NB2. .."..Q.#: F7
0001E0  00 0E 00 00 23 00 0E 00 00 23 00 02 00 00 23 00 :....#....#....#.: 87
0001F0  09 00 00 4E 2F 10 B0 00 67 53 20 4E 2F 20 00 10 :...N/.O.gS N/ ..: CD
000200  61 FE 20 4E 48 60 42 61 74 10 67 B0 00 67 B0 00 :aß NH'Bat.gO.gO.: CA
000210  6D B0 00 6F C0 00 B0 00 6D B0 00 6E 5F C0 00 52 :mO.o§.O.mO.n_§.R: F8
000220  E9 82 51 FF 20 7E 9E 44 00 60 B4 00 66 44 00 60 :i.Q. ß.D.'4.fD.': 59
000230  44 00 4C 00 4E 48 61 48 60 48 61 48 2F E0 61 20 :D.L.NHaH'HaH/'a : B0
000240  2F E8 61 20 C0 00 80 00 B0 00 6F 5E 10 4E 48 C0 :/ha §...O.o^.NH§: BB
000250  22 00 0F 10 C0 00 E1 10 32 24 B2 00 67 B2 00 66 :"...§.a.2$2.g2.f: 79
000260  B0 00 6D B0 00 6E 12 60 C0 00 B2 66 4E 4C 06 4E :O.mO.n.'§.2fNL.N: 73
000270  48 F0 61 FD 22 00 0F 24 00 04 26 00 05 74 20 72 :Hpaü"..$..&..t r: 20
000280  76 10 51 FF 20 61 FF 70 61 00 53 67 51 FF 42 20 :v.Q. a.pa.SgQ.B : 93
000290  61 FD 61 FD 4A 66 32 00 04 20 24 00 0F 34 00 E2 :aüaüJf2.. $..4.b: 0B
0002A0  64 61 5A 51 FF E6 08 00 67 61 5A 08 00 67 61 5A :daZQ.f..gaZ..gaZ: A9
0002B0  61 02 00 30 61 FF 42 20 61 FC 61 FD 4C 0F 4E 30 :a..Oa.B aöaüL.NO: E9
0002C0  00 10 51 FF 5B 4E 2F 72 53 10 51 FF 22 4E 90 00 :..Q.ÄN/rS.Q."N..: 5D
0002D0  60 90 00 60 70 60 70 60 70 60 70 48 60 61 FC 42 :'..'p'p'p'pH'aöB: 77
0002E0  12 E5 20 00 05 22 00 04 24 00 0F 74 10 10 52 51 :.e .."..$..t..RQ: AC
0002F0  FF 20 10 61 FE 70 61 42 20 00 05 61 FC 20 00 05 :. .aßpaB ..aö ..: 48
000300  61 FC 61 FE 66 23 10 20 00 04 42 61 FC 60 61 FD :aöaßf#. ..Baö'aü: D6
000310  4C 07 4E 48 E0 26 00 05 61 FD 66 22 23 00 04 61 :L.NH'&..aüf"#..a: 62
000320  FD 66 90 63 01 E8 60 70 22 00 04 22 61 FC 20 74 :üf.c.h'p".."aö t: 48
000330  48 20 20 61 FE 70 61 FF 10 61 FE 70 61 FF 51 FF :H  aßpa..aßpa.Q.: 46
000340  70 61 FF 4C 02 10 00 10 B0 00 6D B0 00 6C 60 70 :pa.L....O.mO.l'p: 47
000350  10 51 FF 10 00 42 20 61 FB 61 FB 61 05 56 FF 23 :.Q...B aäaäa.V.#: 68
000360  00 04 60 00 48 E0 26 00 05 61 FD 69 00 22 00 04 :..'.H'&..aüi."..: A4
000370  66 22 61 FB 20 20 61 FD 70 61 FE 10 61 FD 70 61 :f"aä  aüpaß.aüpa: 90
000380  FE 42 20 61 FB 20 00 05 7A 61 FB 61 FC 66 53 E2 :ßB aä ..zaäaöfSb: AF
000390  7C DC 13 70 E0 51 FF D3 42 60 69 4A 67 52 60 10 :öö.p'Q.SB'iJgR'.: 5C
0003A0  B0 00 66 42 52 10 B0 00 67 4A 67 12 60 B0 00 66 :O.fBR.O.gJg.'O.f: 0A
0003B0  53 60 B0 00 67 61 FC 60 FF 61 FC 61 FB 23 00 04 :S'O.gaö'.aöaä#..: 66
0003C0  4C 0F 4E 48 F0 26 00 05 61 FC 66 00 24 22 00 04 :L.NHp&..aöf.$"..: 19
0003D0  61 FC 66 53 E2 7C DC 13 70 E0 51 FF D3 60 68 0C :aöfSböö.p'Q.S'h.: AA
0003E0  00 66 12 0C 00 67 4A 66 20 00 04 26 96 32 01 C1 :.f...gJf ..&.2.A: 6F
0003F0  34 53 20 53 66 61 03 66 32 01 B5 66 51 FF 61 FA :4S Sfa.f2.5fQ.az: 23
000400  20 00 05 20 90 61 FC 42 20 00 05 61 FA 60 61 FB : .. .aöB ..az'aä: B0
000410  61 FA 4C 0E 4E 61 FB 66 23 00 04 60 69 08 00 00 :azL.Naäf#..'i...: BD
000420  04 22 00 04 34 67 53 24 33 00 34 4E 51 FF 20 00 :."..4gS$3.4NQ. .: 61
000430  04 4E 4C FF 00 04 2F 00 04 3F 00 04 4E 61 FB 4E :.NL.../..?..NaäN: 0F
000440  61 FB 66 33 00 04 60 69 42 00 04 08 00 00 04 60 :aäf3..'iB......': 74
000450  61 FB 4E 48 7F 00 04 4E 23 00 04 33 00 04 23 00 :aäNH...N#..3..#.: 44
000460  04 23 00 04 2E 00 06 61 F9 61 FC 53 00 04 6B F9 :.#.....ayaöS..ky: D1
000470  60 FF 48 E0 12 02 00 0C 00 67 00 22 00 04 61 FB :'.H'.....g."..aä: 90
000480  66 08 00 42 32 0C 00 67 E5 D2 22 E4 52 22 00 04 :f..B2..geR"dR"..: 8A
000490  32 60 69 34 67 53 42 20 00 05 61 F9 30 52 61 FB :2'i4gSB ..ayORaä: 88
0004A0  10 00 70 61 FC 20 61 FB 42 20 00 05 61 F8 51 FF :..paö aäB ..axQ.: 69
0004B0  60 20 00 10 61 F8 60 61 FA 4C 06 4E 4A 61 FA 22 :' ..ax'azL.NJaz": 0B
0004C0  00 04 34 67 53 94 6D E5 D3 22 00 51 FF 53 00 04 :..4gS.meS".Q.S..: 74
0004D0  60 48 7F 00 04 4E 23 00 04 33 00 04 23 00 04 55 :'H...N#..3..#..U: 53
0004E0  00 04 23 00 04 2E 00 06 22 00 04 34 67 53 24 34 :..#....."..4gS$4: CB
0004F0  00 51 FF 61 F8 20 00 11 61 F8 61 FA 60 F8 48 80 :.Q.ax ..axaz'xH.: AE
```

Bild 4. Der Monitor, gerade Adressen

```
000500  61 FA 66 22 61 F9 66 90 63 24 61 F9 66 C1 B5 6E :azf"ayf.c$ayfA5n: 5E
000510  14 53 66 60 D3 D5 15 53 66 60 61 F9 4C 06 4E 48 :.Sf'SU.Sf'ayL.NH: 45
000520  C0 61 F9 67 61 F9 60 22 61 F9 66 F9 B0 63 F9 22 :§aygay'"ayfyOcy": 44
000530  61 F9 66 F9 0C 00 6F 00 00 6F 22 B2 6C 60 12 B2 :ayfy..o..o"2l'.2: 13
000540  6C 60 32 B2 6C 4C 02 4E 48 FF 00 04 20 00 11 60 :l'22lL.NH... ..': 94
000550  48 FF 00 04 20 00 11 46 20 61 F7 61 F7 4E 23 00 :H... ..F awawN#.: 03
000560  04 20 00 23 00 04 61 F9 33 00 00 04 20 00 11 61 :. .#..ay3... ..a: 6E
000570  F7 20 00 61 F9 61 F7 61 F9 60 F7 48 00 20 00 11 :w .ayaway'wH. ..: F3
000580  60 48 00 20 00 11 60 48 00 20 00 11 60 48 00 20 :'H. ..'H. ..'H. : 7A
000590  00 11 60 48 00 20 00 11 60 48 00 20 00 11 60 48 :..'H. ..'H. ..'H: 6B
0005A0  00 20 00 11 46 20 61 F7 61 F6 4C 01 48 FF 00 04 :. ..F awavL.H...: DE
0005B0  4E 23 00 04 20 00 23 00 04 33 00 00 04 61 F8 61 :N#.. .#..3...axa: AD
0005C0  F6 61 F9 60 F6 48 41 42 12 3E 30 00 4E 4A 00 04 :vay'vHAB.>O.NJ..: 8D
0005D0  67 32 30 00 4E 4C 00 4E 2F 32 00 30 00 4E 4A 67 :g2O.NL.N/2.O.NJg: 41
0005E0  22 4E 2F 32 00 30 00 4E 4A 67 30 00 22 4A 4E 08 :"N/2.O.NJgO."JN.: F2
0005F0  00 00 04 67 20 00 10 61 F6 4E 20 00 0F 60 20 00 :...g ..avN ..' .: EF
000600  02 10 00 11 00 00 22 00 00 4E 33 00 00 04 30 00 :......"..N3...O.: FA
000610  4E 18 61 F7 42 0C 00 00 66 12 61 0C 00 67 B2 00 :N.awB...f.a..g2.: 0A
000620  00 67 52 54 60 B8 00 00 67 22 00 01 61 01 61 4A :.gRT'8..g"..a.aJ: BC
000630  67 20 00 10 60 F5 42 00 04 4E 20 00 10 60 F5 04 :g ..'uB..N ..'u.: 09
000640  00 6C 06 00 0C 00 6E 0C 00 6D 4E 12 00 4E 48 70 :.l....n..mN..NHp: CB
000650  12 54 26 34 00 42 54 51 FF 22 34 00 12 00 54 51 :.T&4.BTQ."4...TQ: B3
000660  FF 61 F7 22 32 00 14 67 0C 00 67 0C 00 6D 04 00 :.aw"2..g..g..m..: 16
000670  12 54 51 FF 60 4A 67 22 D3 00 00 32 00 14 67 0C :.TQ.'Jg"S..2..g.: 75
000680  00 6D 04 00 12 54 51 FF 60 42 60 30 00 4C 02 4E :.m...TQ.'B'O.L.N: F5
000690  61 F6 66 F6 4A 00 04 66 00 22 00 01 61 01 42 00 :avfvJ..f."..a.B.: 2E
0006A0  42 00 22 30 00 32 01 4E B0 00 67 30 00 32 01 4E :B."O.2.NO.gO.2.N: DD
0006B0  32 01 20 00 01 C1 61 00 C1 24 30 00 4E 4A 66 61 :2. ..Aa.A$O.NJfa: EA
0006C0  00 20 60 20 00 10 61 F4 20 00 05 20 53 61 F6 42 :. ' ..at .. SavB: 36
0006D0  20 00 05 61 F4 61 F4 4E 20 00 10 60 F4 20 00 10 : ..atatN ..'t ..: D1
0006E0  60 F4 61 F6 66 F6 24 61 F6 66 F6 B0 63 F6 28 32 :'tavfv$avfvOcv(2: 41
0006F0  01 30 00 4E 30 00 4E 0C 00 67 22 00 01 30 00 4E :.O.NO.N..g"..O.N: 11
000700  22 00 01 61 42 00 42 00 22 61 32 01 24 61 30 00 :"..aB.B."a2.$aO.: 73
000710  4E 4A 66 22 B8 6C 30 00 4E 4E 20 00 10 60 F4 20 :NJf"8lO.NN ..'t : B4
000720  00 0E 60 F4 36 00 12 54 51 FF 4E D3 D3 00 00 4E :..'t6..TQ.NSS..N: 90
000730  36 00 12 54 51 FF 4E 13 00 02 20 00 02 11 00 E0 :6..TQ.N... ....': 62
000740  10 11 00 E0 11 00 11 00 13 00 00 02 0C 00 00 00 :...'............: 44
000750  66 22 00 00 42 4E 00 00 48 70 61 10 E1 10 00 4C :f"..BN..Hpa.a..L: 7E
000760  03 4E 00 0A 00 0A 00 0A 00 0B 00 0B 00 0B 00 0B :.N..............: 9B
000770  00 08 00 0B 00 0B 00 0B 00 0B 58 00 04 44 00 06 :..........X..D..: DA
000780  51 00 07 53 00 06 47 00 08 54 00 08 42 00 08 4D :Q..S..G..T..B..M: F3
000790  00 09 5A 00 0A 45 00 0B 50 00 0B 46 00 0C 52 00 :..Z..E..P..F..R.: BC
0007A0  0D 57 00 0D 00 00 04 44 00 05 41 00 05 53 00 05 :.W.....D..A..S..: 5C
0007B0  55 00 05 53 00 05 50 00 05 00 00 04 44 3D 44 3D :U..S..P.....D=D=: 0D
0007C0  44 3D 44 3D 44 3D 44 3D 44 3D 44 3D 41 3D 41 3D :D=D=D=D=D=D=A=A=: 02
0007D0  41 3D 41 3D 41 3D 41 3D 41 3D 53 3D 55 3D 53 3D :A=A=A=A=A=S=U=S=: 28
0007E0  50 3D 43 59 20 46 20 5A 52 20 45 20 45 54 20 52 :P=CY F ZR E ET R: EB
0007F0  20 53 50 20 4D 3D 0D 50 69 74 72 6F 20 21 20 0A : SP M=.Pitro ! .: F3
000800  0D 50 69 74 72 6F 66 21 20 0A 0D 4E 20 72 61 70 :.Pitrof! ..N rap: 8A
000810  69 74 20 65 20 0D 00 0A 72 65 75 61 20 69 65 61 :it e ...reua iea: 95
000820  65 21 0A 0D 53 65 69 69 64 64 69 65 6E 74 61 6C :e!..Seiiddiental: 6C
000830  77 64 21 0A 0D 4E 20 69 65 61 65 73 65 69 69 64 :wd!..N ieaeseiid: 23
000840  21 0A 0D 46 6C 20 6F 20 6F 6E 20 0D 00 0A 61 20 :!..Fl o on ...a : 2E
000850  20 0D 44 72 63 6F 79 66 6C 20 0D 00 0A 69 6B 66 : .Drcoyfl ...ikf: 71
000860  6C 20 0D 00 0A 2A 2A 20 4D 2D 38 45 42 4D 6E 74 :l ...** M-8EBMnt: 7F
000870  72 56 72 2E 20 2E 20 20 39 35 38 20 2A 2A 0D 00 :rVr. .  958 **..: 1D
000880  3F 3F 0D 00 72 61 70 69 74 68 74 0D 00 63 65 73 :??..rapitht..ces: CF
000890  61 72 73 3A 00 75 20 72 6F 20 74 00 64 65 73 65 :ars:.u ro t.dese: 2B
0008A0  72 72 61 20 69 6C 67 6C 6F 2D 6F 65 61 20 64 76 :rra ilglo-oea dv: D8
0008B0  73 6F 20 79 30 61 20 63 65 6B 65 72 72 61 20 6F :so yOa cekerra o: 98
0008C0  65 66 6F 20 72 6F 20 74 00 72 76 6C 67 64 69 73 :efo ro t.rvlgdis: CA
0008D0  72 63 69 6E 61 20 68 74 31 31 2D 70 6F 65 61 20 :rcina ht11-poea : 5D
0008E0  68 74 31 31 2D 70 6F 65 61 20 00 00 00 00 00 00 :ht11-poea ......: 30

Prüfsumme insgesamt : 023BAA
```

Franz Schomacher

MS-DOS-Format unter CP/M 3.0

Wie man einzelne Sektoren unter CP/M 3.0 auf der Diskette lesen und schreiben kann, wurde in MC 3/85 beschrieben. Die im folgenden Beitrag vorgestellten Programme dienen zum Übertragen ganzer Dateien unter diesem Betriebssystem auf bzw. von MS-DOS-formatierten Disketten.

Will man Dateien von einer MS-DOS-Diskette lesen oder auf diese schreiben, so muß man natürlich wissen, wie Disketten unter MS-DOS organisiert sind: Zunächst einmal gibt es vier verschiedene Formate, nämlich einseitig mit acht Sektoren pro Spur, doppelseitig mit acht Sektoren pro Spur und beides noch einmal mit neun Sektoren pro Spur. Dies bereitet schon die erste Schwierigkeit, denn unter CP/M 3.0 kann man neun Sektoren pro Spur nicht lesen. Deshalb wird bei neun Sektoren eine Datei möglicherweise nur zum Teil übertragen; dies wird aber vom Programm festgestellt und gemeldet.

Format-Erkennung

Um zu erkennen, in welchem Format eine Diskette formatiert wurde, wie auch für die Berechnung der sogenanten Gruppen, braucht man die „File Allocation Table" (FAT), denn in dieser Tabelle stehen die Informationen über das Diskettenformat und die Belegung der Diskette. Für das Format ist immer das erste Byte der Tabelle zuständig, und zwar entspricht:

FFh doppelseitig mit acht Sektoren pro Spur,

FEh einseitig mit acht Sektoren pro Spur,

FDh doppelseitig mit neun Sektoren pro Spur und

FCh einseitig mit neun Sektoren pro Spur.

In der FAT stehen ausserdem die Gruppennummern, die jeweils einem Bereich auf der Diskette zugeordnet sind. Steht hier der Wert 000, so ist die betreffende Gruppe frei und kann benutzt werden. FF8h...FFFh bedeutet, daß es sich um die letzte Gruppe einer Datei handelt. Jede andere Zahl gibt die nächste Gruppe einer Datei an.

Dateien

Die Verkettung dieser Gruppen zu einer kompletten Datei läuft nach folgendem Schema ab: Die Startgruppe einer Datei steht im Inhaltsverzeichnis (Directory).

Will man aus einer Gruppennummer die nächste Gruppennummer bestimmen, so muß zunächst ein Offset in der FAT bestimmt werden, der sich aus der ganzzahligen Multiplikation der alten Nummer mit 1,5 ergibt, denn jeder Eintrag ist 1,5 Byte lang. Dann liest man aus der FAT mit diesem Offset ein Wort (= 2 Byte). War die alte Gruppennummer gerade, so bilden die niederwertigen 12 Bit die neue Gruppennummer, sonst die höherwertigen. Ergibt sich als Gruppennum-

```
 1: CONST Blocksize  = 512; (physikalische Blockgroesse unter MS-DOS )
 2:       BufGr      =  86; (86 fuer COM-File, sonst 24              )
 3: (*********  Achtung BufGr muss eine gerade Zahl sein!   *********)
 4:                        (Gilt so nur fuer Commodore PC128         )
 5: TYPE  DPB = RECORD                          (Disk-Parameter-Block)
 6:                SPT                 :Integer; (vgl. mc Heft 3/85   )
 7:                BSH, BLM, EXM       :Byte;
 8:                DSM, DRM            :Integer;
 9:                AL0, AL1            :Byte;
10:                CKS, OFF            :Integer;
11:                PSH, PHM            :Byte;
12:           END;
13:       tDisk = (doppelacht, einsacht, doppelneun, einsneun);
14:       tmode = (Suche, Zeigen);
15:       tBuff = ARRAY[1..Blocksize] OF Byte;
16:       String12 = STRING[12];
17:       Anystring = STRING[255];
18:
19: VAR   Psector, Psize,
20:       SekpGr, LSekpTr, TrFak,
21:       Eintraege, i, StartGruppe,
22:       Datstart, benutzt             : Integer;
23:       ptr                           : ^DPB ABSOLUTE i;
24:       DPH                           : ^Integer ABSOLUTE i;
25:       Diskart                       : tDisk;
26:       FATBuf                        : ARRAY[1..2] OF tBuff;
27:       DatBuf                        : ARRAY[1..BufGr] OF tBuff;
28:       Suchname, CPMName                   : String12;
29:       sel                           : Byte;
30:       Dateiende, erst, gefunden,
31:       Abbruch                       : Boolean;
32:       Antwort                       : Char;
33:
34: PROCEDURE Meldung(Meldetext: Anystring);
35: (Dient zur Ausgabe einer Meldung in der Menuzeile 23              )
36: BEGIN
37:   GotoXY(1,23);ClrEol;
38:   Write(#27,'G4');              (Schaltet auf inverse Darstellung  )
39:   Write(#7,' ',Meldetext,' ');
40:   Write(#27,'G0');              (Schaltet zurueck auf normale Darstellung)
41:   Write('  ')
42: END; (Meldung)
43:
44: FUNCTION Ubios(FN, PA, PBC, PDE, PHL: Integer): Integer;
45: (Da unter CP/M 3.0 BIOS-Funktionen ueber BDOS-Funktion 50 aufgerufen)
46: (werden, wird diese Funktion benutzt. vgl. mc Heft 3/85        )
47: TYPE ParameterBlock = RECORD
48:                         func, Areg          :Byte;
49:                         BCreg, DEreg, HLreg :Integer;
50:                       END;
```

Bild 1. Mit diesem Programm können Dateien von MS-DOS-formatierten Disketten gelesen werden

```
51: VAR BiosPB         :ParameterBlock;
52:     Result         :Integer;
53: BEGIN
54:    WITH BiosPB DO
55:    BEGIN
56:       func  := FN;
57:       Areg  := PA;
58:       BCreg := PBC;
59:       DEreg := PDE;
60:       HLreg := PHL;
61:    END;
62:    Result := 0;
63:    CASE FN OF
64:       2,3,7,13,14,15,17,18,19,24: Result := Bdos(50, Addr (BiosPB));
65:       9,16,20,22,25:              Result := BdosHL(50, Addr (BiosPB));
66:       ELSE                        Bdos(50, Addr (BiosPB));
67:    END;
68:    Ubios := Result;
69: END; (Ubios)
70:
71: PROCEDURE Init;
72: BEGIN
73:   i := Ubios(9, 0, 0, 0, 0);           {Laufwerk -hier 0 = aktuell-}
74:   Bdos(14,0);                          {auswaehlen.                }
75:   i := BdosHl(31);                     {Adresse des DPB ermitteln. }
76:   Psize := 1 SHL ptr^.PSH;
77: END; (Init)
78:
79: PROCEDURE RWSektor(Dr, Tr, lSec : Integer; Schreib: Boolean;
80:                    VAR SecBuf : tBuff);
81: { Liest einen physikalischen Sektor von Diskette in SecBuf,          }
82: { bzw. schreibt SecBuf auf Diskette, je nach Wert von Schreib.       }
83: BEGIN
84:    lSec := lSec DIV Psize;                 {log. in phys. Sektor    }
85:    i := Ubios (9, 0, Dr, 1, 0);            {Drive auswaehlen        }
86:    Psector := Ubios (16, 0, lSec, DPH^, 0);{phys. Sektor berechnen  }
87:    i := Ubios (23, 0, 1, 0, 0);            {nur einen Sektor lesen  }
88:    i := Ubios (10, 0, Tr, 0, 0);           {Spur waehlen            }
89:    i := Ubios (11, 0, Psector, 0, 0);      {Sektor waehlen          }
90:    i := Ubios (12, 0, Addr(SecBuf), 0, 0); {DMA-Adresse waehlen     }
91:    i := Ubios (28, 1, 0, 0, 0);            {Bank waehlen            }
92:    IF Schreib
93:    THEN i := Ubios (14, 0, 0, 0, 0)        {phys. Sektor schreiben  }
94:    ELSE i := Ubios (13, 0, 0, 0, 0)        {phys. Sektor lesen      }
95: END;{RWSektor}
96:
97: PROCEDURE LiesFAT;
98: {Liest die File-Allocation-Table in einen Buffer und setzt je nach }
99: {Diskart (einseitig -doppelseitig bzw. neun - acht Sektoren) Werte }
100: {fuer Directoryeintraege und zur Umrechnung von Gruppen in Track   }
101: {und Sektor fest.                                                 }
102: BEGIN
103:   RWSektor(0,0,4,false,FATBuf[1]);
104:   Diskart := tDisk(Abs(FatBuf[1,1] - $ff));
105:   IF Ord(Diskart) > 1
106:   THEN RWSektor(0,0,8,false,FATBuf[2]);
107:   IF Diskart IN [doppelacht, doppelneun]
108:   THEN SekpGr := 2
109:   ELSE SekpGr := 1;
110:   IF Diskart IN [doppelacht, einsacht]
111:   THEN LSekpTr := 32
112:   ELSE LSekpTr := 36;
113:   IF Diskart IN [einsacht, einsneun]
114:   THEN BEGIN
115:          TrFak := 2;
116:          Eintraege := 64
117:        END
118:   ELSE BEGIN
119:          TrFak := 1;
120:          Eintraege := 112
121:        END
122: END; (LiesFAT)
123:
124: PROCEDURE Datenstart_bestimmen;
125: (Setzt je nach Diskart den Wert fuer Datenstart fest.)
126: BEGIN
127:   CASE Diskart OF
128:     doppelacht: Datstart := 40;
129:     einsacht:   Datstart := 28;
130:     doppelneun: Datstart := 48;
131:     einsneun:   Datstart := 36
132:   END
133: END; (Datenstart_bestimmen)
134:
135:
136: PROCEDURE Gruppe_in_Track_Sektor(Gr: Integer; VAR Tr, Sec: Byte);
137: (Berechnet aus der Gruppennummer Spur und Sektor auf der Diskette.)
138: VAR weiter     : Char;
139: BEGIN
140:   Gr := Gr - 2;
141:   Gr := Gr * SekpGr;
142:   Gr := Gr * 4;
143:   Gr := Gr + DatStart; (Start der Daten)
144:   Tr := Gr DIV LSekpTr;
145:   Sec := Gr MOD LSekpTr;
146:   IF Sec > 31
147:   THEN BEGIN
148:          Meldung('Der folgende Sektor kann nicht gelesen werden. '
149:                  + 'Weiter? (j/n)');
150:          Read(KBD,weiter); Write(weiter);
151:          Abbruch := Upcase(weiter) = 'N'
152:        END;
153:   Tr := Tr * TrFak;
154: END; (Gruppe_in_Track_Sektor)
155:
156: PROCEDURE Nameneingeben(VAR Name: String12);
157: (Dient zur Eingabe des Namens der zu uebertragenden Datei     )
158:   PROCEDURE Mache_Gross(VAR Str: String12);
159:   (Wandelt alle Buchstaben in Str in Grossbuchstaben um       )
160:   VAR i : Integer;
161:   BEGIN
162:        FOR i := 1 TO 12 DO
163:        IF Str[i] <> ' ' THEN Str[i] := Upcase(Str[i])
164:   END;(Mache_Gross)
165:
166: BEGIN
167:   Meldung('Bitte Name der zu uebertragenden Datei angeben (Typ) ');
168:   Read(Name);
169:   CPMname := Name;
170:   Name := Copy(Name,1,Pos('.',Name)-1)+Copy(Name,Pos('.',Name)+1,3);
171:   IF Length(Name) < 11
172:   THEN Name := Copy(Name,1,Length(Name)-3)
173:                +Copy('          ',1,11-Length(Name))
174:                + Copy(Name,Length(Name)-2,3);
175:   Mache_Gross(Name);
176: END; (Nameneingeben)
```

```
 1: PROGRAM Schreib_MS_DOS_Disketten;
 2: (Entwickelt auf Commodore PC 128 unter CP/M+ mit einem Disketten-)
 3: (laufwerk Commodore 1571                                        )
 4:
 5: ($i diskproc.inc)
 6:
 7: VAR AnzSektoren : Integer;
 8: PROCEDURE DirectoryEintrag(Gr: Integer);
 9: (Dient zum Eintrag von Namen und Startgruppe in die Directory der)
10: (MS-DOS-Diskette                                                 )
11: VAR DirBuf                          : tBuff;
12:     DirTr, DirSec, Zaehl, ZeilZaehl,
13:     EinZaehl, k                     : Byte;
14:     Offset                          : Integer;
15:     Ende                            : Boolean;
16:     Vglname                         : String12;
17: BEGIN
18:   DirTr := 0;
19:   DirSec := 12;
20:   RWSektor(0,DirTr,DirSec,false,DirBuf);
21:   gefunden := False;
22:   Ende := False;
23:   EinZaehl := 0;
24:   WHILE  NOT Ende AND NOT gefunden DO
25:   BEGIN
26:     Offset := 0;
27:     REPEAT
28:       EinZaehl := Succ(EinZaehl);
29:       IF (DirBuf[Offset+1] = 0) OR (DirBuf[Offset+1] = $e5)
30:             THEN BEGIN (Freier Platz gefunden)
31:                   FOR k := 1 TO 11 DO
32:                       DirBuf[Offset+k] := Ord(Suchname[k]);
33:                   FOR k := 12 TO 26 DO
34:                       DirBuf[offset+k] := 0;
35:                   gefunden := True
36:                 END;
37:       IF NOT gefunden
38:       THEN BEGIN
39:              Ende := (DirBuf[Offset+1] = 0) OR (EinZaehl = Eintraege);
40:              Offset := Offset + 32
41:            END
42:     UNTIL gefunden OR Ende OR (Offset >= 512);
43:     IF NOT gefunden AND NOT Ende
44:     THEN BEGIN
45:            DirSec := Dirsec + 4;
46:            RWSektor(0,DirTr,DirSec,false,DirBuf)
47:          END
48:   END; (While) (Platz gefunden oder Directory zuende)
49:   IF gefunden
50:   THEN BEGIN
51:          DirBuf[Offset+27] := Lo(Gr);
52:          DirBuf[Offset+28] := Hi(Gr);
53:          RWSektor(0,DirTr,DirSec,True,DirBuf)
54:        END
55:   ELSE BEGIN
56:          Meldung('Directory voll');
57:          Halt
58:        END
59: END; (DirectoryEintrag)
60:
```

```
61: PROCEDURE SchreibeDatei;
62: (Schreibt die Datei auf die MS-DOS-Diskette und aktualisiert die FAT.)
63: VAR k, z, Gruppe              : Integer;
64:     Track, Sektor             : Byte;
65:     Start, oben               : Boolean;
66:
67: FUNCTION FreieGruppe(k :Integer; VAR ob :Boolean): Boolean;
68: (Dient zur Ueberpruefung, ob in der FAT an der Stelle k eine freie)
69: (Gruppe existiert, und zwar werden in Abhaengigkeit von ob die    )
70: (oberen (ob = true) oder unteren (ob = false) 12 Bits untersucht. )
71: VAR Hilf : Integer;
72:     h    : Boolean;
73: BEGIN
74:   h := False;
75:   IF ob
76:   THEN BEGIN (obere Bits untersuchen (von k und k+1))
77:          Hilf := FATBuf[1,k+1] * 256 + FATBuf[1,k];
78:          IF Hilf AND 4095 = 0 THEN h := True
79:        END;
80:   IF NOT h
81:   THEN BEGIN (noch keine freie Gruppe gefunden,                )
82:              (also untere 12 Bits untersuchen (von k+1 u. k+2))
83:          Hilf := FATBuf[1,k+2] * 256 + FATBuf[1,k+1];
84:          IF Hilf SHR 4 = 0
85:          THEN BEGIN h := true; ob := False END
86:        END;
87:   FreieGruppe := h
88: END; (FreieGruppe)
89:
90: PROCEDURE Trageein(k :Integer; ob :Boolean);
91: (Dient dazu, eine Gruppe in der FAT als belegt mit dem entsprechenden )
92: (Wert zu kennzeichnen.                                                 )
93: VAR Ein, Hilf, j : Integer;
94:     h           : Boolean;
95: BEGIN
96:   j := k;
97:   IF Dateiende
98:   THEN Ein := $FFF
99:   ELSE BEGIN (naechste freie Gruppe suchen und eintragen)
100:         h := NOT ob;
101:         IF h THEN j := j + 3;
102:         WHILE NOT FreieGruppe(j,h) AND (j < 512) DO
103:               j := j + 3;
104:         IF j < 512
105:         THEN BEGIN
106:                IF h THEN Ein := Trunc((j-1)/1.5)
107:                     ELSE Ein := Trunc((j-1)/1.5)+1
108:              END
109:         ELSE BEGIN
110:                Meldung('Diskette voll');
111:                Halt
112:              END;
113:       END;
114:   IF ob
115:   THEN BEGIN
116:          Hilf := FATBuf[1,k+1] *256 + FATBuf[1,k];
117:          Hilf := Hilf OR Ein;
118:          FATBuf[1,k+1] := Hi(Hilf);
119:          FATBuf[1,k]   := Lo(Hilf)
120:        END
```

Bild 2. Zum Schreiben auf MS-DOS-formatierte Disketten mit diesem Programm dürfen nur acht Sektoren pro Spur vorhanden sein

```
121:   ELSE BEGIN
122:          Hilf := FATBuf[1,k+2] *256 + FATBuf[1,k+1];
123:          Ein := Ein SHL 4;
124:          Hilf := Hilf OR Ein;
125:          FATBuf[1,k+2] := Hi(Hilf);
126:          FATBuf[1,k+1] := Lo(Hilf)
127:        END
128: END;(Trageein)
129:
130: BEGIN
131:   k := 4; z := 1; Start := True; Dateiende := False;
132:   WHILE NOT Dateiende DO
133:   BEGIN
134:     oben := True;
135:     WHILE NOT FreieGruppe(k,oben) AND (k < 512) DO
136:           k := k + 3;
137:     IF k >= 512
138:     THEN BEGIN
139:            Meldung('Diskette voll');
140:            Halt
141:          END;
142:     IF oben THEN Gruppe := Trunc((k-1)/1.5)
143:             ELSE Gruppe := Trunc((k-1)/1.5) + 1;
144:     IF Start
145:     THEN BEGIN
146:            Directoryeintrag(Gruppe);{Startgruppe in Directory eintragen}
147:            Start := False
148:          END;
149:      Gruppe_in_Track_Sektor(Gruppe,Track,Sektor);
150:      RWSektor(0,Track,Sektor,True,DatBuf[z]);
151:      IF Diskart = doppelacht
152:      THEN BEGIN {bei doppelseitigen Disketten zweiten Sektor schreiben}
153:             Sektor := Sektor + 4;
154:             z := z+1;
155:             Track := Track + Sektor DIV LSekpTr;
156:             Sektor := Sektor MOD LSekpTr;
157:             RWSektor(0,Track,Sektor,True,DatBuf[z])
158:           END;
159:      Dateiende := z >= AnzSektoren;
160:      Trageein(k,oben);
161:      z := z+1
162:   END;
163:   {FatBuf auf Diskette schreiben}
164:   RWSektor(0,0,4,True,FatBuf[1]);
165:   RWSektor(0,0,8,True,FatBuf[1]); {in Sektor 8 steht das Doppel der FAT}
166: END; {SchreibeDatei}
167:
168: PROCEDURE LiesDatei;
169: {Liest die Datei von der CP/M-Diskette und bestimmt dabei }
170: {die Anzahl der Sektoren fuer die MS-DOS-Diskette.        }
171: {Fehlermeldung, falls Datei zu gross                      }
172: VAR Anzahl : Integer;
173:     Hilf : FILE;
174: BEGIN
175:   Assign(Hilf,CPMName);
176:   Reset(Hilf);
177:   Anzahl := Filesize(Hilf);
178:   IF Anzahl > BufGr*4
179:   THEN BEGIN
180:          Meldung('Datei zu gross! Buffer vergroessern. <COM-File!>');
181:          Halt
182:        END;
183:    BlockRead(Hilf,DatBuf,Anzahl);
184:    IF Anzahl MOD 4 = 0
185:    THEN AnzSektoren := Anzahl DIV 4
186:    ELSE AnzSektoren := Anzahl DIV 4 + 1 ;
187:  END; {LiesDatei}
188:
189: BEGIN {Hauptprogramm}
190:   ClrScr;
191:   REPEAT
192:       Meldung('Bitte CP/M-Diskette einlegen und <RETURN>');
193:       Readln;
194:       Init;
195:       Nameneingeben(Suchname);
196:       LiesDatei;
197:       Meldung('MS-DOS-Diskette einlegen und <RETURN>');
198:       Readln;
199:       Init;
200:       sel := 1;
201:       LiesFAT;
202:       IF Diskart IN [doppelneun, einsneun]
203:       THEN BEGIN
204:              Meldung('Achtung neun Sektoren pro Spur!'
205:                      + ' schreiben nicht moeglich');
206:              Halt
207:            END;
208:     Datenstart_bestimmen;
209:     SchreibeDatei;
210:     Meldung('Weitere Dateien uebertragen? <j/n> ');
211:     Read(KBD,Antwort); Write(Antwort);
212:     Antwort := Upcase(Antwort)
213:   UNTIL Antwort = 'N'
214: END.
```

```
 1: {$r+}
 2: PROGRAM Lies_MS_Dos_Disketten;
 3: {Entwickelt auf Commodore PC 128 unter CP/M+ mit einem Disketten-}
 4: {laufwerk Commodore 1571                                -----     }
 5:
 6: {$idiskproc.inc}
 7:
 8: PROCEDURE ScanDirectory(Art: tmode);
 9: {Dient je nach Wert von Art zur Namensuche oder zur Ausgabe}
10: {des Inhaltsverzeichnisses der MS-Dos-Diskette             }
11: VAR DirBuf                          : tBuff;
12:     DirTr, DirSec, Zaehl, ZeilZaehl,
13:     EinZaehl                        : Byte;
14:     Offset                          : Integer;
15:     Ende                            : Boolean;
16:     Vglname                         : String12;
17: BEGIN
```

Bild 3. Diese Include-Datei wird von den Programmen in Bild 1 und Bild 2 benötigt

```
18:    DirTr := 0;
19:    CASE Diskart OF
20:      doppelacht, einsacht: DirSec := 12;
21:      doppelneun, einsneun: Dirsec := 20
22:    END;
23:    RWSektor(0,DirTr,DirSec,False,DirBuf);
24:    gefunden := False;
25:    Ende := False;
26:    ZeilZaehl := 1;
27:    EinZaehl := 0;
28:    WHILE  NOT Ende AND NOT gefunden DO
29:    BEGIN
30:      Offset := 0;
31:      REPEAT
32:        EinZaehl := Succ(EinZaehl);
33:        FOR Zaehl := 1 TO 11 DO
34:            VglName[Zaehl] := Chr(DirBuf[Offset + Zaehl]);
35:        CASE Art OF
36:        Suche:  gefunden := Copy(VglName,1,11) = Copy(Suchname,1,11);
37:        Zeigen: IF (DirBuf[Offset+1] <> 0) AND (DirBuf[Offset+1] <> $e5)
38:                THEN BEGIN
39:                       Write(Copy(VglName,1,8),' ',Copy(VglName,9,3),' ');
40:                       IF ZeilZaehl MOD 6 = 0 THEN Writeln;
41:                       ZeilZaehl := Succ(ZeilZaehl)
42:                     END
43:        END;
44:        IF NOT gefunden
45:        THEN BEGIN
46:               Ende := (DirBuf[Offset+1] = 0) OR (EinZaehl = Eintraege);
47:               Offset := Offset + 32
48:             END
49:      UNTIL gefunden OR Ende OR (Offset >= 512);
50:      IF NOT gefunden AND NOT Ende
51:      THEN BEGIN
52:             DirSec := Dirsec + 4;
53:             RWSektor(0,DirTr,DirSec,False,Dirbuf)
54:           END
55:    END; (While) (Name gefunden oder Directory zuende)
56:    IF gefunden
57:    THEN StartGruppe := DirBuf[Offset+28] * 256 + DirBuf[Offset+27]
58:    ELSE IF Art <> Zeigen
59:         THEN BEGIN
60:                Meldung('Datei nicht gefunden <RETURN>');
61:                Readln
62:              END
63: END; (ScanDirectory)
64:
65:
66: PROCEDURE Lies;
67: (Liest die Datei von der MS-DOS-Diskette in einen Datenbuffer bis)
68: (zum Dateiende oder bis der Buffer voll ist.                      )
69:
70:   FUNCTION NeueGruppe(AGr: Integer): Integer;
71:   (Berechnet aus der alten Gruppennummer den Offset in der)
72:   (FAT und die neue Gruppennummer.                         )
73:   VAR Offset, Hilf : Integer;
74:   BEGIN
75:     Offset := Trunc(AGr * 1.5)+1;
76:     IF Offset > 512
77:     THEN BEGIN
78:            Offset := Offset - 512;
79:            sel := 2
80:          END;
81:     Hilf := FATBuf[sel,Offset+1] * 256 + FATBuf[sel,Offset];
82:     IF Odd(AGr) THEN NeueGruppe := Hilf SHR 4
83:                 ELSE NeueGruppe := Hilf AND 4095
84:     (Falls alte Gruppe ungerade obere 12 Bits, )
85:     (sonst untere 12 Bits als neue Gruppe      )
86:   END; (NeueGruppe)
87:
88: VAR z, Gruppe    : Integer;
89:     Track, Sektor : Byte;
90:     Ende          : Boolean;
91: BEGIN
92:   IF NOT erst
93:   THEN BEGIN
94:          Meldung('MS-DOS-Diskette einlegen und <RETURN>');
95:          Readln;
96:          Init
97:        END;
98:   Gruppe := Startgruppe;
99:   z := 1;
100:  Ende := False;
101:  WHILE NOT Ende AND (z <= BufGr) AND NOT Abbruch DO
102:  BEGIN
103:    Gruppe_in_Track_Sektor(Gruppe,Track,Sektor);
104:    IF NOT Abbruch
105:    THEN RWSektor(0,Track,Sektor,False,DatBuf[z]);
106:    IF (Diskart = doppelneun) OR (Diskart = doppelacht)
107:    THEN BEGIN
108:           z := z+1;
109:           Sektor := Sektor +4;
110:           Track := Track + (Sektor DIV LSekpTr);
111:           IF Sektor >= LSekpTr
112:           THEN Sektor := Sektor - LSekpTr;
113:           IF NOT Abbruch
114:           THEN RWsektor(0,Track,Sektor,False,DatBuf[z])
115:         END;
116:    z := Succ(z);
117:    Gruppe := NeueGruppe(Gruppe) ;
118:    IF (Gruppe >=  $ff8) AND (Gruppe <= $fff)
119:    THEN Ende := True
120:  END; (While)
121:  Dateiende := Ende;
122:  Startgruppe := Gruppe;
123:  benutzt := z - 1;
124: END; (Lies)
125:
126: PROCEDURE Schreib;
127: (Schreibt den Inhalt des Datenbuffers auf die CP/M-Diskette bis)
128: (zum Dateiende oder bis der Buffer leer ist.                   )
129: VAR z : Integer;
130:     help : FILE;
131: BEGIN
132:   Meldung('CP/M-Diskette einlegen und <RETURN> '); Readln;
133:   Init;
134:   Assign(Help,CPMName);
135:   IF erst
136:   THEN BEGIN
```

mer ein Wert zwischen FF8h und FFFh, so ist die Datei zu Ende. Bei einem anderen Wert muß aus der so erhaltenen Gruppennummer eine logische Sektornummer berechnet werden, indem zunächst 2 von der Gruppennummer subtrahiert wird (die erste Gruppe überhaupt ist Gruppe 2). Das Ergebnis wird mit der Anzahl von Sektoren pro Gruppe multipliziert. Nachdem dann noch die logische Sektornummer des Datenstarts hinzuaddiert wurde, hat man die logische Sektorennummer, aus der sich nun Spur und Sektor auf der Diskette durch Division und Restbildung ermitteln lassen. Bei einseitig beschriebenen Disketten muß die Spurnummer ausserdem noch mit 2 multipliziert werden, da ungerade Spuren immer auf der Rückseite der Diskette gesucht werden.

```
137:          Rewrite(Help);
138:          erst := False
139:        END
140:   ELSE BEGIN
141:          Reset(Help);
142:          Seek(Help,FileSize(Help))
143:        END;
144:   BlockWrite(Help,Datbuf,4 * benutzt);
145:   Close(Help)
146: END; {Schreib}
147:
148: PROCEDURE Directory_ja_nein;
149: {Fragt ab, ob das Inhaltsverzeichnis gezeigt werden soll }
150: BEGIN
151:   Meldung('Soll das Inhaltsverzeichnis gezeigt werden? <j/n> ');
152:   Read(KBD,Antwort); Write(Antwort);
153:   IF Upcase(Antwort) = 'J'
154:   THEN BEGIN
155:          ClrScr;
156:          ScanDirectory(Zeigen)
157:        END
158: END; {Directory_ja_nein}
159:
160: BEGIN {Hauptprogramm}
161:   ClrScr;
162:   Meldung('Bitte MS-DOS-Diskette einlegen und <RETURN>');
163:   Readln;
164:   Init;
165:   sel := 1;
166:   LiesFAT;
167:   IF Diskart IN [doppelneun, einsneun]
168:   THEN BEGIN
169:          Meldung('Achtung neun Sektoren pro Spur!'
170:                  + ' Leseversuch dennoch starten? <j/n>');
171:          Read(KBD,Antwort); Write(Antwort);
172:          Abbruch := Upcase(Antwort) = 'N'
173:        END
174:   ELSE Abbruch := False;
175:   WHILE NOT Abbruch DO
176:   BEGIN
177:     Directory_ja_nein;
178:     REPEAT
179:       Nameneingeben(Suchname);
180:       ScanDirectory(Suche);
181:       IF NOT gefunden AND (Upcase(Antwort) <> 'J')
182:       THEN Directory_ja_nein
183:     UNTIL Gefunden;
184:     Datenstart_bestimmen;
185:     Dateiende := False;
186:     erst := True;
187:     WHILE NOT Dateiende AND NOT Abbruch DO
188:     BEGIN
189:       Lies;
190:       IF NOT Abbruch
191:       THEN schreib
192:     END;
193:     Meldung('Weitere Dateien uebertragen? <j/n> ');
194:     Read(KBD,Antwort); Write(Antwort);
195:     IF Upcase(Antwort) = 'J'
196:     THEN BEGIN
197:            Meldung('MS-DOS-Diskette einlegen und <RETURN>');
198:            Readln;
199:            Init
200:          END;
201:     Abbruch := Upcase(Antwort) = 'N'
202:   END
203: END.
```

Jeder Eintrag im Directory besteht aus 32 Byte. Am Anfang steht der Dateiname oder, wenn das Inhaltsverzeichnis zu Ende ist, der Wert 00h. Die Startgruppe der Datei steht in den Bytes 26 und 27 in der übliche Form, also Low-Byte vor High-Byte. Die restlichen Einträge sind in diesem Zusammenhang nicht wichtig.

Die Programme

Im Programm „Lies–MS-DOS-Disketten" (*Bild 1*) wird nach der Initialisierung und dem Einlesen der FAT in einen Puffer zunächst nach dem Namen der zu übertragenden Datei gefragt. Wird die Datei nicht gefunden, kann man sich das Inhaltsverzeichnis zeigen lassen. Ist die Suche nach der Datei erfolgreich verlaufen, so wird diese in einen Puffer gelesen und nach der Aufforderung zum Diskettenwechsel unter demselben Namen auf die CP/M-Diskette geschrieben. Dieser Vorgang wird solange wiederholt, bis die ganze Datei übertragen ist. In der gleichen Art kann man dann weitere Dateien übertragen.

Es empfiehlt sich, das Programm nach dem Testen in eine COM-Datei zu compilieren, da dann der Puffer für die Daten grösser werden kann. Alles weitere kann den Kommentaren im Programm entnommen werden.

Auf MS-DOS-Disketten schreiben kann man mit dem Programm „Schreib–MS-DOS-Disketten" (*Bild 2*). Voraussetzung für das korrekte Beschreiben einer MS-DOS-Diskette mit diesem Programm ist, daß die Diskette auf einem MS-DOS-Rechner mit acht Sektoren pro Spur formatiert wurde, sonst kann sie unter CP/M-Plus nicht beschrieben werden. Das Programm läuft nun wie folgt ab: Nach Einlegen der Diskette, auf der die zu übertragenden Dateien stehen, wird nach dem Namen einer Datei gefragt.

Nach Eingabe des Namens und Einlesen der Datei muß die Diskette gewechselt werden, um die Datei auf die MS-DOS-Diskette zu schreiben. Dabei wird Sektor für Sektor auf die Diskette geschrieben, die FAT aktualisiert und der Name in das Inhaltsverzeichnis eingetragen. Im Gegensatz zum Leseprogramm darf die Datei nicht größer sein als der Datenpuffer, aber man kann die Datei bei Bedarf ja aufteilen.

In dem Include-File „Diskproc.inc" (*Bild 3*) stehen die Deklarationen, die für beide Programme gleich sind.

Stefan Hammes

Submit unter OS-9/68k

Umsteiger von CP/M-68k werden unter OS-9 die Submit-Funktion vermissen. Wenn es unter diesem Betriebssystem auch weit elegantere Methoden gibt, so ist doch Vertrautes gerade am Anfang wichtig. Darum zeigt der folgende Beitrag, wie man die Submit-Funktion in C realisieren kann.

Für Leser, denen Submit kein Begriff ist, zunächst eine kurze Erklärung: Es kommt häufig vor, daß man wiederholt Kommandofolgen mit gleichen oder ähnlichen Argumenten eingeben muß, zum Beispiel beim wiederholten Compilieren eines Programmes. Das ist nicht nur langweilig, es können sich auch leicht Tippfehler einschleichen. Hier schafft das CP/M-Kommando Submit Abhilfe. Man legt eine Textdatei an, die die auszuführenden Kommandos enthält. In dieser Datei können Platzhalter für Argumente stehen, die erst beim Aufruf übergeben werden.

Beim Aufruf von Submit mit dem Namen der Textdatei und den besagten Argumenten nimmt einem dieses Programm dann die ganze Tipparbeit ab, indem es die Platzhalter durch die Argumente ersetzt und die so gewonnenen Kommandos ausführt.

Kenner von OS-9 werden jetzt vielleicht sagen: So etwas gibt es doch unter dem Namen „cfp". Das stimmt zwar, jedoch ist die Funktion cfp nur in der Lage, ein Argument zu bearbeiten. Noch schwerer wiegt, daß cfp aus den resultierenden Kommandos eine Datei erzeugt, die dann verarbeitet wird, als ob sie von der Standard-Eingabe käme. Das hat zur Folge, daß nach dem Aufruf eines Texteditors der Rest dieser Datei vom Editor als Text-Eingabe gewertet und entsprechend behandelt wird, anstatt nach Verlassen des Editors den weiteren Ablauf zu steuern.

Diese Nachteile hat das hier vorgestellte Programm Submit nicht. Es ist in C geschrieben und sollte daher, eventuell nach geringen Änderungen, auch unter Unix lauffähig sein.

Die Bedienung

Zunächst erstellt man eine Textdatei mit der Endung ".sub". In dieser können beliebige Betriebssystem-Kommandos auftreten. Als Platzhalter wird das $-Zeichen, gefolgt von einer Ziffer (1...9) benutzt. Dabei korrespondieren die Ziffern mit der Position der Argumente. Ist zu einer Ziffer kein Argument vorhanden, so wird an dieser Stelle eine leere Zeichenkette eingesetzt. Als Beispiel mag *Bild 1* dienen: Dort wird zunächst ein Editor mit einem Quellprogramm als Argument aufgerufen. Nach Verlassen des Editors wird es compiliert und schließlich gesichert. Beim ersten Aufruf fehlt das zweite Argument ($2), daher wird es durch eine leere Zeichenkette ersetzt. Beim zweiten Aufruf hat das zweite Argument den Wert „-g" und wird entsprechend ersetzt.

Die Funktion des Programmes ist durch die Kommentare dokumentiert (*Bild 2*). Ein paar Worte sollten vielleicht noch hinzugefügt werden: Die Funktion „Index" liefert einen Zeiger auf das erste Auftreten eines Zeichens in einer Zeichenkette, und eine Null, wenn das Zeichen nicht darin enthalten ist. Die Funktion „ersetze" verarbeitet die Submit-Datei zeilenweise, findet mit Hilfe von Index die Platzhalter und ersetzt sie durch die entsprechenden Argumente aus der Kommandozeile. Dies wird rekursiv so lange fortgesetzt, bis keine Platzhalter mehr auftreten. Nach der Eingabe einer Kommandozeile in der Form
SUBMIT <prog> <a1> <a2> ... <an>
hat das Zeichenketten-Feld Argv folgende Inhalte:

argv[0] = <prog>
argv[1] = <a1>
argv[2] = <a2>
...
argv[n] = <an>

Die Variable argc enthält die Zahl der Argumente plus Eins. Die von der Funktion „ersetze" bearbeiteten Zeichenketten werden auf dem Bildschirm zur Kontrolle ausgegeben und danach der Funktion „system" übergeben. Diese Funktion betrachtet die empfangene Zeichenkette als Shell-Kommando, führt es aus und suspendiert für die Zeit dieser Ausführung das Submit-Programm.
Das Programm kann leicht nach eigenen Vorstellungen erweitert werden. Es wäre denkbar, mehr Optionen einzubauen, Kommentarzeilen zuzulassen oder die Zahl der möglichen Argumente zu erhöhen.

Literatur:

[1] *Brian W. Kernighan, Dennis Ritchie:* Programmieren in C, Carl Hanser Verlag, München, ISBN 3-446-13878-1
[2] Microware Systems Corp.: OS-9/68000 C-Compiler User's Manual, Microware, Des Moines, Iowa

```
* Das ist das SUBMIT-File 'c.sub'.

edt $1.c
cc $1.c -ix -m=10 $2
copy $1.c /d1/cpgms/$1.c -vr

* Folgende Kommandos werden nach 'submit c test' ausgefuehrt.

edt test.c
cc test.c -ix -m=10
copy test.c /d1/cpgms/test.c -vr

* Ergebnis bei 'submit c test -g' (-g erzeugt Symboltable fuer Debugger).

edt test.c
cc test.c -ix -m=10 -g
copy test.c /d1/cpgms/test.c -vr
```

Bild 1. Hier einige Anwendungs-Beispiele des Submit-Programmes

```
#include <stdio.h>

#define LINE_LEN    256

char    help_txt[] =

"Syntax: submit [<opts>] [<submit file>] [<<parameter>>]\n\
Function: Executes commands from the submit file with textual replacement.\r
          The submit file must have the suffix .sub.\n\
Options:\n\
     -?         displays this help message\n";

/*
    Bei 'fgets' wird das Zeichen '\n' mitgelesen.
    Mit dieser Funktion wird es geloescht.
*/
kill_lf(s)
char        s[];
{
    s[strlen(s)-1] = '\0';
}

/*
    Diese Funktion ersetzt im String 's' die auftretenden
    '$' Zeichen mit entsprechenden Argumenten aus der Kommandozeile.

*/
ersetze(s,argc,argv)
char    *s;
char    *argv[];
int     argc;
{
    int nummer;
    char    *position;
    char    temp[LINE_LEN];
    char    *index();

    /*
        Falls kein '$' auftritt, ist man fertig.
    */
    if ((position=index(s,'$')) == NULL) return;
    /*
        Also gibt es doch eins, daher Ersetzung noetig.
    */
    strncpy(temp,s,(int)(position - s));    /* 1. Teil des Strings */
    /*
        Es muss ein '\0' Zeichen angefuegt werden,
        da 'strncpy' das nicht macht.
    */
    temp[(int)(position - s)] = '\0';
    /*

        Hier wird die Zahl, die hinter '$' steht, extrahiert.
    */
    nummer = *(position+1)-'0';
    /*
        Wenn die Zahl groesser als die Argumentzahl ist, ignorieren.
    */
    if ((nummer > argc-2) || (nummer < 1))
        strcpy(position,position+2);        /* Bei zu grossem Argument */
    else
    {
        strcat(temp,argv[nummer+1]);        /* Textersatz */
        strcat(temp,position+2);        /* Rest des Strings */
        strcpy(s,temp);             /* Ursprungsstring zurueck */
    }
    /*
        Rekursiv auf weitere '$' untersuchen.
    */
    ersetze(s,argc,argv);
}

/*
    Hauptprogramm.
*/
main(argc,argv)
int argc;
char    *argv[];
{
    FILE    *submit;
    FILE    *fopen();
    char    line[LINE_LEN];

    if (argc < 2)
    {
        puts("Zu wenig Parameter");
        puts(help_txt);
        exit(1);
    }
    if (!strcmp(argv[1],"-?"))  /* Help-Message angefordert */
    {
        puts(help_txt);
        return;
    }
    /*
        SUBMIT-File oeffnen (Endung .sub wird generiert).
    */
    strcpy(line,argv[1]);
    strcat(line,".sub");
    if ((submit=fopen(line,"r")) == NULL)
    {
        puts("Fehler beim Oeffnen des SUBMIT-Files");
        puts(help_txt); exit(1);
    }
    /*
        Jeweils eine Zeile aus dem SUBMIT-File lesen.
    */
    while (fgets(line,LINE_LEN,submit) != NULL)

    {
        if (strlen(line) < 2) continue;    /* Kann nicht gueltig sein */
        kill_lf(line);              /* '\n' entfernen */
        ersetze(line,argc,argv);        /* Text-Ersatz durchfuehren */
        puts(line);                 /* Command auf Console */
        system(line);               /* und ausfuehren */
    }
    fclose(submit);
}
```

Bild 2. Mit diesem Programm steht die Submit-Funktion auch unter OS-9 zur Verfügung

Dieter Strauß

1-MBit-EPROMs programmieren

Universelles Programmiergerät pp-100

Der PROM-Programmer pp-100 von Merlin Elektronik ist ein universelles Programmiergerät, mit dem fast alle derzeit erhältlichen EPROMs und EEPROMs programmiert werden können. mc hat den pp-100 getestet.

Der pp-100 wird mit der Programmierbox, einer PC-Erweiterungskarte, einer Diskette mit der Betriebssoftware und einem Handbuch in Deutsch geliefert (*Bild 1*). Zum Betrieb wird ein IBM-PC/XT/AT oder ein kompatibler Computer mit mindestens 256 KByte RAM und PC-MS-DOS 2.0 oder höher benötigt. Beim Auspacken dieser Teile fällt als erstes die etwas ungewohnt aussehende, aber dennoch sehr funktionell gestaltete Programmierbox auf. Sie ist mit zwei Nullziehkraftfassungen ausgestattet und hat oberhalb dieser Fassungen eine Ablagefläche aus leitfähigem Kunststoffschaum. Somit können auch empfindlichste CMOS-EPROMs nach dem Programmieren sicher auf dieser antistatischen Unterlage abgelegt werden. Die PC-Interfacekarte belegt die I/O-Adressen 300H bis 308H. Ein anderer Adreßbereich kann eingestellt werden, falls diese Adressen bereits durch eine andere Karte belegt sind.

Die Verbindung der PC-Erweiterungskarte zur Programmierbox wird durch ein ungefähr ein Meter langes 34poliges Flachbandkabel hergestellt. Die Programmierbox wird über das Flachbandkabel mit Strom versorgt. Die Stromaufnahme ist mit 100 mA bei 12 V und 300 mA bei 5 V gering. In der Programmierbox werden mit einem Spannungswandler und nachgeschalteten Spannungsreglern die verschiedenen Programmierspannungen erzeugt. Der Hardwareaufbau macht insgesamt einen guten Eindruck.

Laut Handbuch wird das Programm mit der Eingabe von PPINSTALL installiert.

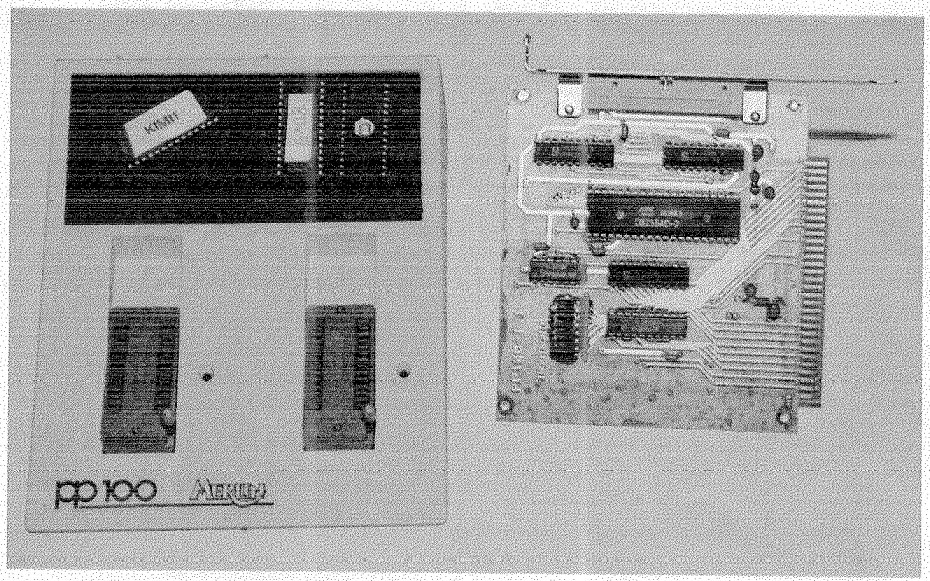


Bild 1. Programmierbox und PC-Erweiterungskarte

Bei unserem Test erschien darauf die kurze Bildschirmmeldung: Bad Command or file name. Nach einem kurzen Blick in das Directory ließ sich das Programm mit INSTALL starten. Als erstes berechnet die Installationsroutine den Leistungsindex des verwendeten PCs. Ein Standard-PC mit 8088 und 4,77 MHz kommt dabei auf den Leistungsindex 5, der Olivetti M24 auf 12 und ein AT-Nachbau auf 18. Nach dem Leistungsindex werden die Bezeichnungen von allen unterstützten EPROMs und EEPROMs auf den Bildschirm ausgegeben. Weiter lassen sich mit der Installationsroutine Standardprogrammiermodus, Landessprache, Peripherie-Adresse und ein Modus zum Betreiben von Grafikkarten ohne „Schneegestöber" auf dem Bildschirm eingeben. Wenn das Installationsprogramm ausgeführt ist, läßt sich das eigentliche Betriebsprogramm mit „PP100" aufrufen. Auf dem Bildschirm erscheint im oberen Fenster das Hauptmenü (*Bild 2*). In der unteren Bildschirmhälfte erscheint in einem zweiten Fenster der voreingestellte EPROM-Typ mit seinen Kenndaten und einigen voreingestellten Parametern wie z. B. Start- und Endadresse des RAM-Puffers und Startadresse des EPROMs. Der Typ des Bausteins wird mit dem zweiten Menüpunkt und die voreingestellten Parameter werden mit dem dritten Menüpunkt (Initialisieren) ausgewählt.

pp-100 auf dem Prüfstand

In der Redaktion haben wir den pp-100 in den Commodore PC-10 und einen Taiwan-PC mit 256 KByte RAM eingebaut. EPROM-Typen diverser Hersteller haben sich einwandfrei pogrammieren lassen. Interessant ist, daß sich mit dem zweiten Menüpunkt (Bausteintyp festlegen) die großen EPROMs wie 27512 und 27011 nicht auswählen ließen. Hatten wir das Programm falsch bedient oder gar die Karte beim Einbauen in den PC beschädigt? Nach etwas nervösem Suchen im Handbuch war die Ursache schnell gefunden. Der 1-MBit-EPROM 27011 braucht – was einfach nachzurechnen ist – 128 KByte Puffer-RAM. Ein so großer Pufferbereich ist in einem PC mit 256 KByte RAM und einem Programm, das über 100 KByte belegt, nicht frei. Das Betriebsprogramm ist so intelligent, daß bei zu kleinem Speicherausbau diese großen EPROMs nicht ausgewählt werden können. Für den unkundigen Bediener wäre es jedoch sicher besser, eine kurze Meldung auf dem Bildschirm erscheinen zu lassen, daß das Puffer-RAM zu klein ist.

In jeder Bildschirmmaske kann mit F10 „Hilfe" angefordert werden. Die jeweilig erscheinenden zusätzlichen Informationen sind genau auf den gerade aktuellen Bedienfall abgestimmt, so daß kein seitenlanges Blättern durch den gesamten Hilfstext notwendig ist.

Unter dem dritten Menüpunkt werden die Programmierparameter initialisiert. Will man mit anderen Parametern als den voreingestellten arbeiten, so lassen sich hier die Konfigurationen der beiden Fassungen, der Programmieralgorithmus, die Start- und die Endadresse des RAM-Puffers und die Startadresse des PROMs eingeben. Vier Fassungskonfigurationen stehen zur Verfügung:

– Baustein in Fassung 1 bearbeiten
– Fassung 1 und 2 parallel
– Fassung 1 und 2 seriell
– Fassung 1 und 2 gesplittet

Im Modus „Fassung 1 und 2 seriell" werden die beiden Bausteine als ein Baustein mit doppelter Kapazität angesehen. Im Modus „Fassung 1 und 2 gesplittet" werden die beiden Bausteine logisch als ein Baustein mit doppelter Datenbusbreite aufgefaßt. Beim Lesen der PROM-Daten werden die Daten des ersten Bausteins auf gerade Adressen und die Daten des zweiten Bausteins auf ungerade Adressen gelegt. Beim Programmieren werden Arbeitspeicherinhalte mit geradzahliger Adresse in den ersten Baustein und solche mit ungeradzahliger Adresse in den zweiten Baustein geschrieben. Diese Konfiguration wird in Anwendungen mit 16-Bit-Mikroprozessoren benötigt.

Unter dem vierten Menüpunkt läßt sich der Pufferspeicher editieren. Der Editor arbeitet im Hex- oder ASCII-Modus. Zwischen den beiden Modi wird mit Ctrl-Cursor rechts und -Cursor links umgeschaltet. Mit der Insert-Taste (Ins) kann ein zweites Bildschirmfenster eröffnet werden. Mit der Esc-Taste wird ein Untermenü aufgerufen, unter dessen Menüpunkten weitere komfortable Editierfunktionen vorhanden sind. Speicherblöcke lassen sich mit Hex-Zahlen oder ASCII-Strings füllen. Blöcke können kopiert, Bytes gesucht und die Prüfsumme eines Speicherblocks gebildet werden. Grundrechenoperationen lassen sich mit Hexadezimal- und Dezimalzahlen durchführen.

Vom pp-100 werden fünf Dateiformate verstanden:
– ASCII-Binär-Format

```
PP100                    Version 2.01 (c) 1986 MERLIN              Hilfe=F10

                          1 Mit Baustein arbeiten
                          2 Bausteintype festlegen
                          3 Initialisierung
                          4 Speicher editieren
                          5 Datenfile lesen
                          6 Datenfile schreiben
                          7 Betriebssystem rufen
                          8 Daten drucken
                          9 Zurück ins System

                    EPROM-Bezeichnung:        2764
                    Programmierspg.  :        12.5 V
                    Kapazität        :        2000
                    EPROM, in Sockel 1 bearbeiten
                    Intel-Intelligent
                    RAM-Start        :        0000
                    RAM-Ende         :        1FFF
                    PROM-Start       :        0000
```

Bild 2. Das Hauptmenü

```
PP100                    Version 2.01 (c) 1986 MERLIN              Hilfe=F10

2716*     2716      27C16*    2732*     2732      27C32*    27C32     2764*
2764      27C64*    27C64     87C64     27128*    27128     27C128*   27C128
27256*    27256     27C256*   27C256    27512     27C512    27513     27011
2532      2564      5143      5133      52B13H    52B13     52B33     DQ2B16A
2816A     2817A     2864A     28C64     2364

                    EPROM-Bezeichnung:        27C256
                    Programmierspg.  :        12.5 V
                    Kapazität        :        8000
                    EPROM, in Sockel 1 bearbeiten
                    Intel-Intelligent
                    RAM-Start        :        0000
                    RAM-Ende         :        7FFF
                    PROM-Start       :        0000
```

Bild 3. Die Bauteilebibliothek des pp-100

– Intel-Hex-Format
– Motorola-2-Byte-Format (S-Record)
– Motorola-3-Byte-Format
– Motorola-4-Byte-Format

Der pp-100 kann diese Formate sowohl lesen als auch schreiben.

Welche Typen können programmiert werden?

Unter dem ersten Menüpunkt wird mit den Bausteinen selbst gearbeitet. Es können PROM-Daten gelesen, Bausteine programmiert, gelöscht (nur EEPROMs) und verifiziert werden. Jedes Byte wird, bevor der Programmierimpuls angelegt wird, ausgelesen und mit dem Inhalt des Pufferspeichers verglichen. Stimmen beide Werte überein, so wird dieses Byte bei der Programmierung übersprungen.

Vor dem Programmieren kann ein Blankcheck durchgeführt werden. Vom System wird dabei geprüft, ob der Baustein wirklich leer ist. Für Tippfaule ist die Reihenfolge Blankcheck, Programmieren und Verifizieren als ein Untermenüpunkt zusammengefaßt. Einfaches Kopieren der Daten vom ersten Baustein zum zweiten Baustein ist auch möglich. In diesem Fall müssen jedoch beide Bausteine vom gleichen Typ sein.

Die pp-100-Bauteilebibliothek umfaßt 26 EPROM-Grundtypen und 11 EEPROM-Typen (*Bild 3*). Vier Programmieralgorithmen werden unterstützt: Standardalgorithmus, Intel-Intelligent, Intel-Quick-Pulse und Fujitsu-Quick-Pro.

Im Anhang des Handbuchs befindet sich ein Bausteinverzeichnis von 21 verschiedenen Herstellern. Für jeden Baustein dieser Hersteller ist ein Grundtyp aus der pp-100-Bausteinebibliothek und der vorgeschlagene Programmieralgorithmus aufgeführt.

All diese Eigenschaften, die früher nur bei wesentlich teureren Systemen zu finden waren, machen den pp-100 zu einem komfortablen und zuverlässigen Programmiergerät.

Peter Pernsteiner

GfA-Basic

Strukturierte Programmierung mit Atari-ST

Basic auf dem 68000 – hat das denn Zukunft? Auch ich hatte vor Beginn dieses Tests ähnliche Gedanken. Doch mußte ich mich eines Besseren belehren lassen: Beim GfA-Basic-Interpreter erinnert eigentlich nur noch der Name an eine langsame, umständliche und unstrukturierte Programmierung.

Lang hat es gedauert, bis für den Atari ST ein voll funktionsfähiges Basic zu haben war. Als dann das ST-Basic erschien, erwartete fast jeder, daß endlich ein Interpreter da ist, mit dem man mehr anstellen kann, als man es von den „alten" 8-Bit-Heimcomputern her gewohnt war. Doch die Enttäuschung war groß; ST-Basic war nur ein Produkt mit vielen Fehlern und einer unbefriedigend schlechten Rechengenauigkeit. Als im Sommer 1986 erstmals GfA-Basic vorgestellt wurde, war dies ein Lichtblick für alle Besitzer eines Atari ST. Die letzten kleinen Fehlerchen und Schwächen der ersten Version wurden jetzt mit der Version 2.0 abgestellt, die gegenüber der ersten Version 30 Befehle mehr enthält. Der nun verfügbare Interpreter mit einem Umfang von nur 56 409 Byte Programm-Code bietet hohe Rechengenauigkeit (11 Stellen) bei ungewohnt hoher Geschwindigkeit (10 000 leere FOR...NEXT-Schleifen in weniger als ½ Sekunde) und hohem Bedienungskomfort.
Neben der Qualität dieses Produkts überzeugt auch der überraschend niedrige Preis von 169 DM.

```
T=Timer!   *** Bench_1a ***
For I=1 To 10000
Next I
Print (Timer-T)/2000;" ms"
0.1145 ms

I=1!       *** Bench_1b ***
T=Timer
Do
  Exit If I=10000
  Inc I
Loop
Print (Timer-T)/2000;" ms"
0.2985 ms

I=1!       *** Bench_1c ***
T=Timer
While I<=10000
  Inc I
Wend
Print (Timer-T)/2000;" ms"
0.3 ms

I=1!       *** Bench_1d ***
T=Timer
Repeat
  Inc I
Until I=10000
Print (Timer-T)/2000;" ms"
0.2745 ms
```

Bild 1. GfA-Basic bietet vier verschiedene Schleifenstrukturen

Ungewohnte Basic-Syntax

Bei GfA-Basic fällt zuerst auf, daß man beim Programmieren keine Zeilennummern vergeben muß und daß je Programmzeile nur ein Befehl erlaubt ist. Programme werden dadurch zwar optisch länger, aber auch besser lesbar, weil der Interpreter das Listing bei offenen Schleifen selbsttätig um zwei Zeichen einrückt. Die Programme werden automatisch, während der Eingabe des Programmtextes, optisch strukturiert aufgebaut. Auch eingefleischte Basic-Programmierer, die es gewohnt sind, mehrere Befehle hintereinander in eine Zeile zu schreiben, können ihre alten Programme mit dem GfA-Basic weiterverwenden. Dazu ist auf der Programm-Diskette eine Routine enthalten, die Basic-Programme von ihren Zeilennummern befreit und Mehrbefehl-Zeilen auftrennt und einrückt. Bei unserem Test haben wir GW-Basic-Listings vom Olivetti M24 auf den Atari ST übertragen und mit der Routine „Stkill" ohne Probleme konvertiert. Alte und bewährte Programme müssen also nicht neu eingegeben werden.
Auf dem Atari ST können sie mit dem GfA-Basic nur noch schneller und komfortabler werden. Mit GfA-Basic wird aber nicht nur strukturiert gelistet, sondern man kann tatsächlich strukturierte Programmierung betreiben: Dies wird durch einige neue Befehle erreicht, die bisher bei kaum einem Basic anzutreffen waren. So gibt es neben Schleifen mit FOR...NEXT- auch solche mit DO...LOOP-, WHILE...WEND- und REPEAT...UNTIL-Strukturen (*Bild 1*). Die im Bild angegebenen Zeiten beziehen sich auf einen Schleifendurchlauf. Besonders erfreulich ist, daß neben den frei definierbaren Formelfunktionen (DEFFN) auch Prozeduren (PROCEDURE) erstellt werden können. Diese lassen sich beliebig mit globalen und lokalen Variablen vereinbaren; ja sie können sich sogar selbst aufrufen (rekursive Prozeduren).
Das alles sind Eigenschaften, die das Herz eines jeden Programmierers höher schlagen lassen. Fast schon selbstverständlich ist in diesem Zusammenhang, daß Variablen-Namen beliebig lang sein dürfen, was für die Verständlichkeit eines Programmes sehr von Vorteil ist. Zur weiteren Dokumentation lassen sich am Ende einer jeden Programmzeile (außer nach DATA) Kommentare anfügen oder per „REM" zwischen Programmzeilen einschieben. Wer bei all diesen Vorzügen den Befehl „GOTO Zeilennummer" vermißt, kann beruhigt werden: GfA-Basic erlaubt den Einsatz von Labels, die per „GOTO Label" angesprungen werden.

```
T=Timer!   *** Bench_2a ***
For I%=1 To 10000
Next I%
Print (Timer-T)/2000;" ms"
0.04 ms

I%=1!      *** Bench_2b ***
T=Timer
Do
  Exit If I%=10000
  Inc I%
Loop
Print (Timer-T)/2000;" ms"
0.2375 ms

I%=1!      *** Bench_2c ***
T=Timer
While I%<=10000
  Inc I%
Wend
Print (Timer-T)/2000;" ms"
0.2405 ms

I%=1!      *** Bench_2d ***
T=Timer
Repeat
  Inc I%
Until I%=10000
Print (Timer-T)/2000;" ms"
0.215 ms
```

Bild 2. Die Schleifendurchläufe werden mit Integervariablen merklich schneller

Variablen, die es in sich haben

GfA-Basic kennt vier verschiedene Typen von Variablen: Integer-, Real-, String- und Bool'sche Variablen.

Realvariablen sind 6 Byte lang und werden als Gleitkommazahlen angezeigt. Der Atari ST rechnet dabei mit 11stelliger Genauigkeit und einem Exponenten bis zu 154. Alle Zahlen können in vielen Variationen formatiert ausgegeben werden: So lassen sich beispielsweise Kommas zur Tausendertrennung einfügen, zwei verschiedene Exponentialformate anwählen und die Position des Dezimalpunktes sowie die Anzeigegenauigkeit festlegen. Wer Speicherplatz und Rechenzeit sparen will, wird sich sehr über die 4-Byte-Integervariablen freuen. Eine leere FOR...NEXT-Schleife ist mit diesem Zahlentyp fast dreimal so schnell wie mit Real-Variablen (*Bild 2*). An dieser Stelle ein Vergleich zu ST-Basic: Für Bench_1a benötigte dieses 0,9 ms pro Schleifendurchlauf – also fast die 8fache Zeit trotz erheblich schlechterer Rechengenauigkeit. Und Bench_2a brachte einen noch größeren Unterschied zutage: 0,6 ms pro Durchlauf bedeutet mehr als das 15fache an Zeitbedarf, obwohl die Integervariablen nur halb so lang wie die von GfA-Basic sind. Es sei mir deshalb verziehen, daß ich im Test keine weiteren Vergleiche mehr zu ST-Basic aufführe. Dafür sei jeder Leser aufgerufen, die Ergebnisse der kleinen Beispielprogramme mit dem von ihm bevorzugten Computer zu vergleichen.

```
A=Ø!      *** Bench_3a ***
T=Timer
For I%=1 To 1ØØØ
  A=A+3.33
Next I%
Print (Timer-T)/2ØØ;" ms"
Ø.245 ms

A=Ø!      *** Bench_3b ***
T=Timer
For I%=1 To 1ØØØ
  Add A,3.33
Next I%
Print (Timer-T)/2ØØ;" ms"
Ø.195 ms

A=1!      *** Bench_4a ***
T=Timer
For I%=1 To 1ØØØ
  A=A*1.11
Next I%
Print (Timer-T)/2ØØ;" ms"
Ø.28 ms
```

```
A=1!      *** Bench_4b ***
T=Timer
For I%=1 To 1ØØØ
  Mul A,1.11
Next I%
Print (Timer-T)/2ØØ;" ms"
Ø.23 ms

A=Ø!      *** Bench_5a ***
T=Timer
For I%=1 To 1ØØØ
  Inc A
Next I%
Print (Timer-T)/2ØØ;" ms"
Ø.145 ms

A%=Ø!     *** Bench_5b ***
T=Timer
For I%=1 To 1ØØØ
  Inc A%
Next I%
Print (Timer-T)/2ØØ;" ms"
Ø.Ø8 ms
```

Bild 4. Für die Addition und Subtraktion gibt es neben dem Standard-Befehl (3_a) noch zwei schnellere Versionen (3_b, 5). Die Multiplikation und Division können ebenfalls merklich beschleunigt werden (4)

Sehr vielseitig sind die Einsatzmöglichkeiten von Stringvariablen, da diese ohne spezielle Dimensionierung bis zu 32 767 Zeichen lang sein dürfen. Jede Stringvariable benötigt dafür sechs Descriptorbytes: die ersten vier stehen für die Adresse des Strings im Arbeitsspeicher und die zwei anderen für die Länge. Mit „PRINT USING" kann man Strings formatiert ausgeben.
Die Stringvariablen spielen bei GfA-Basic nicht nur für die Textspeicherung eine Rolle; sie dienen auch zur Speicherung von Pixel-Images ganzer Bildschirmseiten oder beliebiger Ausschnitte. Das so gespeicherte Bitmuster kann jederzeit an einem beliebigen Ort wieder ausgegeben werden. 16 verschiedene Modi eröffnen hierbei sehr vielfältige Variationen (*Bild 3*).
Alle vier Variablentypen lassen sich in Feldern anordnen. Eindimensionale Felder sind lediglich durch die Größe des Hauptspeichers begrenzt; mehrdimensionale Felder dürfen insgesamt 65 535 Elemente besitzen, was aber für die meisten Anwendungen mehr als genug sein dürfte.

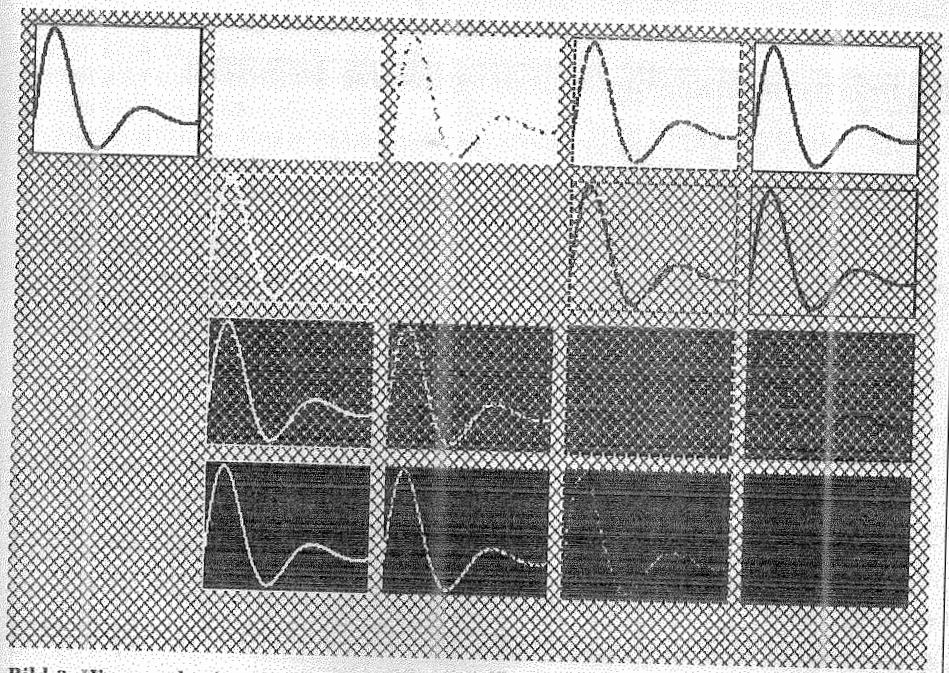


Bild 3. Hier wurde ein Bildschirmausschnitt mit einer gedämpften Sinus-Funktion in 16 verschiedenen Kombinationen mit dem Hintergrundmuster verknüpft

Rechenoperationen mit Pfiff

Die Grundrechenarten sind bei den meisten Basic-Interpretern seit eh und je gleichgeblieben, obwohl es auch andere Verfahren gibt, um zum Beispiel zwei Zahlen zu addieren. Will man beispielsweise den Wert einer Variablen A um eine Zahl (z. B. 3,33) erhöhen, so kann man bei GfA-Basic statt „A=A+3.33" auch „ADD A,3.33" schreiben. Die Ausführungszeit wird durch die Nähe zur Maschinensprache, wie Bench_3a und _3b (*Bild 4*) zeigen, um ein gutes Stück kürzer. Diesen zeitsparenden Befehl gibt es aber nicht nur für die Addition und Subtraktion, sondern auch noch für die Division und Multiplikation (Bench_4): sie laufen erfreulicherweise fast genauso schnell ab, wie die zuvorgenannten.

Will man einen Indexzähler in einer LOOP oder einer ähnlichen Struktur um „eins" erhöhen oder vermindern, so gibt es eine noch elegantere und flottere Möglichkeit: die Inkrement- und Dekrementbildung, die sonst nur in C- und in

```
Deffn Rad(Grad)=Grad*Pi/180
Print "Sin(45)=";Sin(Fn Rad(45))
Sin(45)=0.707106781187
```

Bild 5. Zur übersichtlicheren Programmgestaltung kann man Formel-Funktionen einsetzen

Maschinensprache-Programmen benutzt wird. (Bench_5a im Vergleich zu Bench_3.) Und benutzt man die Inkrementbildung bei Integervariablen, so kann man noch mehr Rechenzeit sparen; die eigentliche Inkrementbildung braucht dann genauso lange, wie ein Durchlauf in einer leeren FOR...NEXT-Schleife (Bench_5b im Vergleich zu Bench_2a).

Fast schon selbstverständlich ist bei einem guten Basic-Dialekt, daß er neben den Grundrechenarten auch trigonometrische Funktionen wie SIN, COS, TAN, ATN (Arcustangens) und Funktionen wie LOG, LOG10 (Zehnerlogarithmus), EXP und SQR beherrscht. Die trigonometrischen Funktionen sind bei GfA-Basic zwar nur im Bogenmaß vorhanden, aber die anderen Winkelmaße können jederzeit hergeleitet und bei häufigerem Bedarf als Formelfunktionen (*Bild 5*) vereinbart werden.

Einen großen Komfort bieten bei GfA-Basic die Zahlenumwandlungsfunktionen: So läßt sich nicht nur der Absolutwert einer Zahl (ABS) und der ganzzahlige Anteil (FIX) bestimmen, sondern auch die Integerfunktion (INT), das Vorzeichen (SGN), der Vorkomma-Anteil (FRAC) und der Nachkomma-Anteil (TRUNC). Ferner kann man prüfen, ob eine Zahl gerade (EVEN) oder ungerade (ODD) ist, und welche Variable innerhalb einer Liste das größte bzw. das kleinste Element darstellt. Zeichenketten lassen sich in numerische Werte umsetzen und umgekehrt. Außerdem können Integervariablen jederzeit als Binär-, Oktal- oder Hexadezimalzahlen dargestellt und in entsprechende Stringvariablen umgesetzt werden. Fast schon selbstverständlich ist in diesem Zusammenhang, daß man Zahlen nicht nur dezimal eingeben kann, sondern auch in den drei anderen Zahlensystemen (durch Vorsatz von „&X", „&O", „&H").

Darüber hinaus verfügt GfA-Basic noch über ein paar Spezialbefehle: Swap, ein wichtiges Element von Forth, gestattet im GfA-Basic den blitzschnellen Austausch der Inhalte zweier Variablen oder ganzer Variablenfelder (*Bild 6*). Bei logischen Operationen kann man neben AND, OR, XOR und NOT auch noch die Implikation (IMP) und die Äquivalenz (EQV) bilden. Für logische Vergleiche gibt es zusätzlich zu den üblichen Vergleichsoperatoren eine auf „ungefähr gleich" (==): Hierzu werden 28 Bit der Mantisse, das sind etwa 8,5 Stellen, miteinander verglichen. Mit den Befehlen „DATE$" und „TIME$" läßt sich direkt vom Basic aus die System-Uhr des Atari ST setzen und eine spezielle Variable „Timer" kann zur Zeitmessung in Schritten einer 200stel Sekunde herangezogen werden.

```
A=1!      *** Bench_6a ***
B=2
T=Timer
For I%=1 To 1000
  Swap A,B
Next I%
Print (Timer-T)/200;" ms"
0.115 ms

A%=1!     *** Bench_6b ***
B%=2
T=Timer
For I%=1 To 1000
  Swap A%,B%
Next I%
Print (Timer-T)/200;" ms"
0.1 ms

A$="1"!   *** Bench_6c ***
B$="2"
T=Timer
For I%=1 To 1000
  Swap A$,B$
Next I%
Print (Timer-T)/200;" ms"
0.13 ms
```

Bild 6. Der Befehl „Swap" erlaubt einen zeitsparenden Austausch von Variablen

Prozeduren mit ungeahnten Fähigkeiten

Nachdem GfA-Basic schon bei seinen Rechenoperationen Erstaunliches leistet, lassen sich durch den gezielten Einsatz von Prozeduren sehr leistungsfähige Programme schreiben. Als Beispiel hierfür sei der von C. A. R. Hoare erfundene rekursive Sortieralgorithmus „Quicksort" aufgeführt. Bisher ließ sich dieser in Basic nur umständlich realisieren. Mit GfA-Basic kann man Quicksort fast

```
Save | Save,A | Quit | New | Blk Sta | Replace | Pg Up | Text 8 | Direct | Run
Load | Merge  | Llist| Block| Blk End | Find    | Pg down| Insert | Flip   | Test
Rem *** Sortier-Test mit Quicksort (rekursiv) ***
Dim A$(999)
Rem                          *** Textfelderzeugung ***
For I%=0 To 999
  For K%=1 To 10
    A$(I%)=A$(I%)+Chr$(65+Random(26))
  Next K%
Next I%
T=Timer
Gosub Quicksort(0,999)!     *** Quicksortaufruf ***
Print "Sortierzeit: ";(Timer-T)/200;" s"
End
Rem    ********************************************
Rem    *** Quicksortprozedur nach C.A.R. Hoare ***
Rem    ********************************************
Procedure Quicksort(Anfang%,Ende%)
  Local A%,Z%
  A%=Anfang%
  Z%=Ende%
  X$=A$((Anfang%+Ende%)/2)
  Repeat
    While A$(A%)<X$
      Inc A%
    Wend
    While X$<A$(Z%)
      Dec Z%
    Wend
    If A%<=Z% Then
      Swap A$(A%),A$(Z%)
      Inc A%
      Dec Z%
    Endif
  Until A%>Z%
  If Anfang%<Z% Then
    Gosub Quicksort(Anfang%,Z%)
  Endif
  If A%<Ende% Then
    Gosub Quicksort(A%,Ende%)
  Endif
Return
```

Bild 7. Dieses Testprogramm erzeugt per Zufall (RANDOM) 1000 Stringvariablen mit je 10 Zeichen Länge. Anschließend wird dieses Stringfeld mit der „Quicksort"-Prozedur sortiert

wortwörtlich vom ursprünglichen Pascal-Programm abschreiben und sogar noch durch die Befehle „SWAP", „DEC" und „INC" weiter vereinfachen und beschleunigen (*Bild 7*). Das Ergebnis ist für einen Interpreter erstaunlich gut: In einer Testreihe wurde zehnmal jeweils per Zufall ein Textfeld mit 1000 Elementen erzeugt und in 7,005 bis 7,595 Sekunden sortiert. Und selbst mit 10 000 Stringvariablen ist GfA-Basic nicht zu erschüttern; es benötigte dafür zwischen 88,425 und 89,515 Sekunden.
GfA-Basic erlaubt nicht nur Basic-Prozeduren: Man kann sogar Unterprogramme in Maschinensprache und mit Parameterübergabe wie in C aufrufen. Außerdem können eigenständige Maschinenprogramme geladen und ausgeführt werden. Sie sehen also, daß den Wünschen der meisten Programmierer mit GfA-Basic kaum mehr Grenzen gesetzt sind.

Mehr als nur Komfort

Ein Basic-Interpreter ist erst dann gut, wenn er auch einen zweckmäßigen Editor hat. Der GfA-Basic-Editor präsentiert sich zwar für passionierte GEM-Anwender etwas ungewohnt (*Bild 7*), ist aber sehr praktisch zu handhaben: Die beiden ersten Zeilen des Bildschirms weisen auf die wichtigsten Editorfunktionen hin, die man alle wahlweise per Funktionstasten aufrufen oder einfach mit der Maus anklicken kann. Neben diesen beiden Zeilen zeigt der Editor normalerweise noch 23 Programmzeilen. Will man sich einen besseren Überblick über das Programm verschaffen und hat man einen SW-Monitor (SM-124) an seinem Atari ST angeschlossen, so werden in einem Kleinschriftmodus sogar 48 Zeilen des Programmlistings auf einmal dargestellt.
Jede Programmzeile kann bis zu 255 Byte lang sein. Beim Eintippen einer längeren Programmzeile wird sinnvollerweise nur diese einzelne Zeile nach links gescrollt; das restliche Listing bleibt linksbündig stehen. Die bereits vorher erwähnte automatische Strukturierung der Programme erfolgt jeweils beim Verlassen der aktuellen Zeile. Dabei wird auch ein Syntax-Check mit entsprechenden Fehlermeldungen durchgeführt.
Durch das Programm-Listing kann man entweder mit den Cursortasten scrollen oder mit den Funktionstasten blättern. Selbstverständlich läßt sich auch jede Position innerhalb des Listings direkt mit der Maus anwählen. Weiß man nicht mehr, wo ein bestimmter Ausdruck in einem längeren Programm steht, findet ihn die schnelle Suchroutine und tauscht ihn auf Wunsch gleich mit einem neuen Ausdruck aus. Innerhalb des gesamten Listings kann man einen Block markieren, der dann grau hinterlegt erscheint und beliebig kopiert, gelöscht oder versetzt werden kann.
Programme können entweder als Programm- oder als ASCII-File gespeichert werden. Zum Speichern und Laden erscheinen die in GEM üblichen Bildschirmmenüs (*Bild 8*). Ein ASCII-File läßt sich an eine Programm-Datei anhängen oder an einem beliebigen Platz einbinden (Merge).
Das Programm-Listing kann direkt auf einem Drucker ausgegeben werden, ohne daß man den Editor verläßt. In einem Direkt-Modus lassen sich Befehle unmittelbar ausführen. Das ist zur Berechnung von mathematischen Ausdrücken oder zum Test einer Basic-Anweisung recht praktisch. Der „Test"-Modus prüft, ob alle Schleifen eines Programm-Textes ordentlich abgeschlossen sind. Will man ein Programm ausführlicher testen, so zeigt ein „Trace"-Modus jeweils vor der Befehlsausführung an, welcher Befehl als nächster kommt.

Grafik und GEM

Anwenderfreundliche Programme mit grafischen Darstellungen, Menüleisten, Infoboxen und Mausabfragen sind etwas sehr Schönes, doch die Programmierung war bisher sehr aufwendig und nicht jedermanns Sache. Mit GfA-Basic wird auch dieses Problem gelöst. Neben vorgefertigten Grafikfunktionen für Linien, Kreise, Ellipsen, Felder, Boxen und so weiter und neben den entsprechenden Füllmuster-, Linienstil- und Linienbreitenroutinen gibt es in GfA-Basic sogar Routinen und Befehle zur Erstellung und Abfrage von Menüleisten, Infoboxen, Mauskoordinaten, Maustasten und vielem mehr. Die Einbindung all dieser Routinen ist, wie man es von Basic erwarten kann, so einfach, daß man schon nach wenigen Versuchen brauchbare Erfolge erzielen kann. Ein in GfA-Basic erstelltes GEM-Programm (*Bild 9*) unterscheidet sich in seinem Äußeren kaum mehr von solchen, die in C oder Modula für den Atari ST erstellt wurden. Damit auch AES-, VDI- und TOS-Funktionen von Basic aus angesprochen werden können, sind entsprechende Befehle in GfA-Basic integriert worden. BIOS-, XBIOS- und GEMDOS-Funktionen können direkt aufgerufen werden.
Für den Anwender gibt es nur noch zwei Unterschiede bei der Anwendung von GfA-Basic-Programmen anstelle von Programmen in anderen Sprachen: Das Programm läuft etwas langsamer als ein compiliertes C- oder Modula-Programm. Vor dem Programmstart muß der Interpreter geladen werden. Das dauert bei Einsatz einer RAM-Disk etwa zwei Sekunden. Von der Diskette wird der Interpreter in etwa zehn Sekunden geladen. Hier muß erwähnt werden, daß die Firma GfA auch an Entwickler gedacht hat, die selbstentwickelte Basic-Programme verkaufen und vermarkten wollen: Im Lieferumfang des GfA-Basic befindet sich zu diesem Zweck eine „RUN-ONLY"-Version des Interpreters, mit der keine Programme geschrieben, sondern nur geladen und gestartet werden können. Dieser Run-only-Interpreter darf

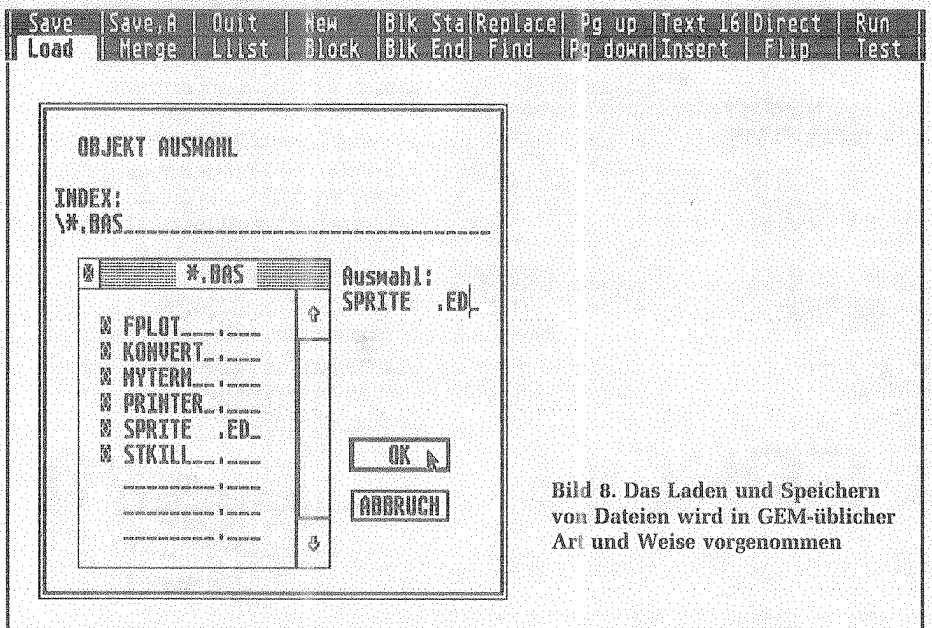
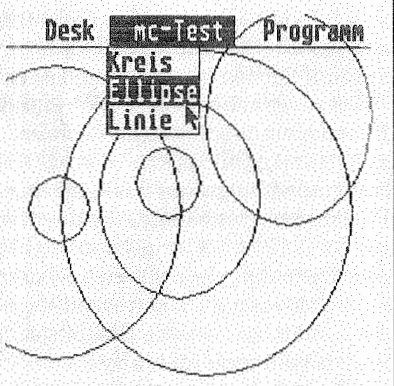


Bild 8. Das Laden und Speichern von Dateien wird in GEM-üblicher Art und Weise vorgenommen

```
| Save   | Save,A | Quit   | New    | Blk Sta | Replace | Pg up  | Text 8 | Direct | Run
| Load   | Merge  | Llist  | Block  | Blk End | Find    | Pg down | Insert | Flip   | Test
Dim Leiste$(50)                          !*** Menüleistendefinition *****
For I=0 To 50
  Read Leiste$(I)
  Exit If Leiste$(I)="***"
Next I
Leiste$(I)=""
Leiste$(I+1)=""
Data Desk,  Info
Data ---------------------
Data 1,2,3,4,5,6,""
Data  mc-Test,Kreis,Ellipse,Linie,""
Data  Programm,Neustart,Ende,""
Data ***
Menu Leiste$()                           !*** Aufbau der Menüleisten ****
On Menu  Gosub Menue
Do
  On Menu                                !*** Menüabfrage ***************
Loop
Rem ***************************************************************************
Rem ************************** Prozeduren ***************************************
Rem ***************************************************************************
Procedure Menue                !*** Menüpunkte aufrufen ******************
  On Menu(0) Gosub Info,0,0,0,0,0,0,0,0,0,Kreis,Ellipse,Linie,0,0,Neustart,Ende
Return
Procedure Ellipse              !*** Ellipsen zeichnen ********************
  Sget Screen$                           !*** Bildschirm speichern ******
  Repeat
    X=Mousex                             !*** Mauskoordinatenabfrage ****
    Y=Mousey
    Vsync
    Sput Screen$
    Line X-50,Y,X+50,Y
    Line X,Y-50,X,Y+50
  Until Mousek=1                         !*** Mausklick abwarten ********
  Repeat
    X1=2*(Mousex-X)
    Y1=2*(Mousey-Y)
    Vsync
    Sput Screen$                         !*** Bildschirm ausgeben *******
    Ellipse X,Y,X1,Y1                    !*** Ellipse zeichnen **********
  Until Mousek=1
Return
Procedure Ende                 !*** Programmende ************************
  Sget Screen$
  M$="Wollen Sie wirklich|schon aufhören?"
  Alert 2,M$,1,"nein|ja",B               !*** Alertbox aufbauen *********
  If B=2
    End


  Endif
  Sput Screen$
Return
Procedure Kreis
  Sget Screen$
  Repeat
    X=Mousex
    Y=Mousey
  Until Mousek=1
  Repeat
    X1=Mousex
    Y1=Mousey
    Vsync
    Sput Screen$
    Circle X,Y,(Sqr((X1-X)^2+(Y1-Y)^2))
  Until Mousek=1
Return
Procedure Linie
  Rem ************************************** Prozedur noch nicht erstellt!
Return
Procedure 0
  Rem ************************************** Systemprozedur **************
Return
Procedure Neustart
  Run
Return
Procedure Info
  Rem ************************************** Info für Vermerke, etc. *****
Return
Rem Ende des Programms
```

Bild 9. Das Listing zeigt, wie einfach man Menüleisten, Zeichenfunktionen und Mausabfragen erstellen und kombinieren kann. Das Demoprogramm ist mit einigen Kreisen ausprobiert worden



mit den selbstgeschriebenen Programmen auch an andere Personen weitergereicht werden.

Dokumentation

Das GfA-Basic ist mit der Run-only-Version und einigen Demo-Programmen auf einer doppelseitig formatierten Diskette, von der sich der Käufer eine Arbeitskopie erstellen kann. Die Diskette wird in einem DIN-A5-Ringbinder geliefert, der ein gut 300 Seiten starkes Handbuch enthält. Im Handbuch befindet sich eine ausführliche Einführung in die Bedienung des Editors und eine sehr ausführliche, alphabetische Befehlszusammenstellung mit erklärenden Beispielen. GfA-Basic ist in Verbindung mit dem ausführlichen Handbuch ein nützliches und vielseitiges Programmier-Werkzeug, das sowohl für Anfänger als auch für Könner zu empfehlen ist.

Ein letzter Hinweis gilt dem angekündigten Compiler: Damit sollen in GfA-Basic erstellte Programme unmittelbar zu einem eigenständigen Programm compiliert werden können. Ob sich die Geschwindigkeit der compilierten Programme tatsächlich um den angekündigten Faktor 10 bis 20 gegenüber dem schon sehr flotten Interpreter beschleunigen läßt, werden wir in einer der nächsten Ausgaben berichten.

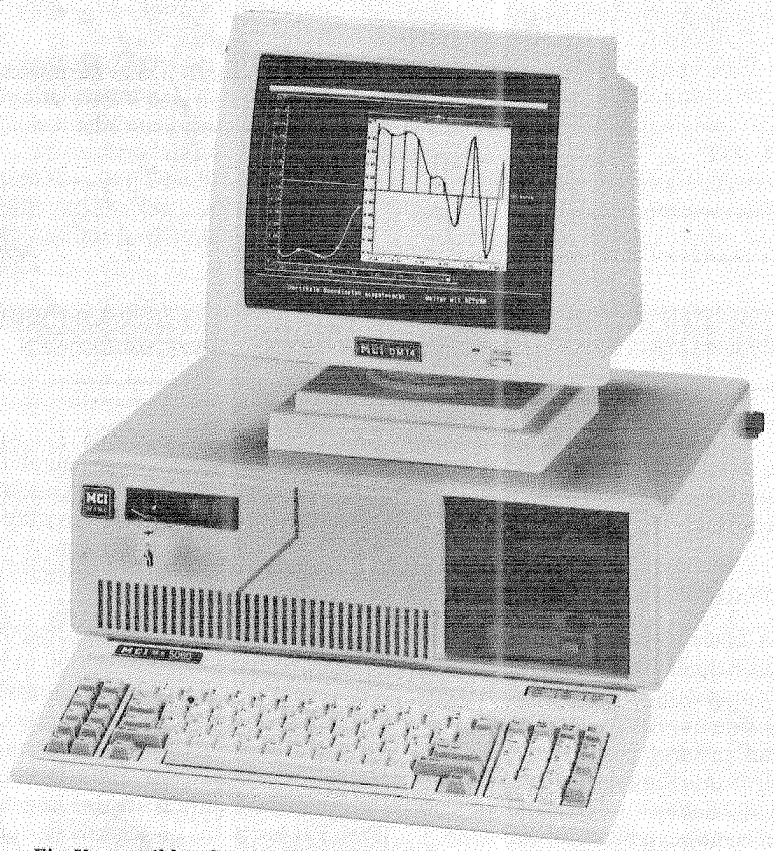
Ulrich Rohde

Preiswerter AT-Kompatibler

Der AT4SLC von MCI

Dieser Test litt unter einem ziemlich großen Mißverhältnis: Der Computer im Test entspricht fortschrittlicher Technologie und ist sehr mächtig, das Handbuch ist unglaublich schmal, es ist ein Handheftchen.

Gleich zu Anfang müssen einige unangenehme Dinge gesagt werden, damit am Schluß das Gute an diesem Modell auch zur Geltung kommen kann. Das erste Unangenehme ist das Verhältnis Handbuch zu Rechner. Das zweite liegt etwas komplizierter. Wenn Sie mc aufschlagen, dann finden Sie viele Angebote preiswerter PC-Kompatibler in den Anzeigen. Und zwar XT- und AT-Kompatible. Diese Rechner sind deshalb so preiswert, weil sie einerseits Kopien sind, die nur mit minimaler eigener Entwicklungsarbeit des Herstellers auf die Welt kamen, andererseits, weil wirklich nur der Rechner allein verkauft wird und kaum Service rundherum. Die Firma MCI hat mir versichert, daß sie alle Garantiefälle zügig abwickeln wird und daß sie bei ihren Produkten so auf Qualität achten würde, daß ein solcher Fall kaum vorkommen würde, denn sonst „würde sich das Ganze nicht rechnen". An dieser Stelle muß man betonen, daß es beim Computerkauf und bei der Inbetriebnahme erfahrungsgemäß zu Situationen kommen kann, in welchen der Betreiber nicht mehr weiter weiß und auf den Service des Herstellers angewiesen ist. In diesen Fällen ist es oft nicht klar, was nun eine Garantieverpflichtung darstellt, was Kulanz ist und wo das Kapitel beginnt, in dem ein Kunde vom Verkäufer mehr will, als es zumutbar ist. Die Firma MCI hat mir versprochen, daß sie hier den Kunden soweit als möglich entgegenkommen würde, hat aber auch gesagt, daß es hier Grenzen gibt.



Ein Kompatibler der AT-Klasse wie der AT4SLC macht das Aufsteigen erschwinglich

Diese Fragen werden hier deshalb so breit erörtert, weil der AT4SLC von MCI ein Rechner ist, der eine Sache für den reiferen Käufer ist. Das soll sagen, daß man aus dem Handbuch des Rechners nicht die Handhabung des nicht im Lieferumfang enthaltenen MS-DOS lernen kann. Dazu wird man auf die Original-Handbücher verwiesen. Damit ist zunächst klar, daß Computeranfänger dieses Modell erst dann kaufen sollten, wenn sie Zugang zu weiterer Literatur haben.

Eine neue Klasse

Von der Festplatte her bootend, meldete sich der Rechner mit dem Signet eines von ATronics lizenzierten ROM-BIOS der Version 3.1. Dieses behauptet, ein speziell für den 80286 entwickeltes BIOS zu sein. Im weiteren Verlauf des Bootens vermeldete mein Rechner, daß er in seinem DOS auch einen MS-DOS-2.1-Treiber eingebaut habe, der das alte PC-Format offenbar erkennen würde. In die Boot-EPROMs eingebaut ist ein Boot-Mechanismus, der schon bei Systemstart zunächst auf der Diskettenstation A: nach einer bootfähigen Diskette sucht und das dort gefundene System hochzieht, egal ob es sich um DOS 2.1 oder 3.x handelt. Es ist also dafür gesorgt, daß Kompatibilität zu den schon auf dem Markt befindlichen PC-Modellen

herrscht. Mein Testexemplar hatte DOS 3.2 auf der Platte, lief aber auch mit 2.11 von der Diskette aus an. Der Beweis für recht weitgehende Kompatibilität unter 3.2 bestand bei mir im ersten Schritt darin, daß ich Euroscript, das ist ein Textverarbeitungsprogramm für PCs, mit Erfolg laden und laufen lassen konnte. Dieser Test ist damit geschrieben. Die Sache mit der Kompatibilität ist aber dennoch nicht so einfach, wie es hier zunächst vermutet werden könnte. Wenn der Rechner von einer 2.1-Diskette bootet, dann kann er das AT-Format der Festplatte nicht erkennen. Der Rechner sollte also tunlichst im 3.x-DOS hochgefahren werden. Außerdem läuft der normale MS-Flugsimulator aufgrund der Konstruktion der ATs prinzipiell nicht.

Neue Laufwerke

Rechner der AT-Klasse haben ein Laufwerk hoher Kapazität eingebaut. Dieses besitzt gegenüber dem alten Standard von 360 KByte Gesamtkapazität eine von 1,2 MByte. Es arbeitet auf zwei Seiten mit je 80 Spuren und einer so hohen Aufzeichnungsdichte, daß normale, doppelseitig beschreibbare Disketten, die für doppelte Aufzeichnungsdichte gefertigt sind, darin nicht mehr zufriedenstellend arbeiten. Man muß hier besondere Disketten mit besonderer Beschichtung verwenden (hohe Koerzitivkraft). Dies gilt allerdings nicht, wenn man MS-DOS-Disketten alter Art benutzen will. Der Rechner liest da alles bereitwillig, was von anderen, einfacheren Modellen kommt und beschreibt dies auch. Nur so konnte ich Euroscript und manches andere auf die eingebaute 20-MByte-Festplatte bringen. Mit dem Beschreiben der Disketten gibt es aber Probleme, wenn man das Ergebnis anderen Rechnern zuführen möchte. Da kann es vorkommen, daß einfachere PCs das Lesen verweigern, weil deren Laufwerke nicht so feine Leseköpfe besitzen wie der AT. Dann werden die „feineren" Signale nicht so präzise aufgefangen, daß es ohne Fehler abgeht. Aber solche Schwierigkeiten müssen nicht in jedem Fall auftreten. Im Test habe ich bei wertvollen Dateien immer darauf geachtet, daß sie auf einer Diskette geschrieben wurden, die vom Zielsystem formatiert wurde. Damit habe ich immer befriedigende Ergebnisse erzielt.
Das Buch, das mir beim Einarbeiten in den AT gut weitergeholfen hat (T. J. Byers: IBM PC AT, Der Rechner der nächsten Generation. McGraw-Hill, Hamburg), schildert, daß noch eine Menge mehr Kompatibilitätsprobleme auftauchen können, wenn man XT- oder PC-Software auf den AT bringen möchte. Zwar klappe das bei etwa 90 % der vorhandenen Software schon, aber Problemfälle seien zum Beispiel Disketten mit Kopierschutz, wenn er laufgeschwindigkeitsabhängig ist (die Disketten rotieren beim AT im Laufwerk schneller), oder wenn er auf inoffiziellen Nebenspuren eines alten Laufwerkes angebracht sei.
Was ist nun so neu am AT, daß er eine andere Klasse von Computern darstellt? Die Antwort lautet: Seine CPU besitzt eine von vornherein auf virtuelle Speicherverwaltung angelegte Logik (die aber auf den alten Standard zurückgeschaltet werden kann), die noch dazu ausgefeilte Multitasking-Techniken durch eingebaute Schutzmechanismen besitzt. Allerdings gibt es in meiner Umgebung dafür keine Software, obwohl der Rechner-Typ in den USA schon länger auf dem Markt ist und auch bei uns seit längerem gekauft werden kann. Auch DOS 3.2 unterstützt den 80286 nicht im Betrieb mit virtuellem Speicher.

Aufwärtskompatibel

Was Intel mit seinen Prozessoren an Strategie verfolgt, wird auf einer Stufe höher bei den Systemen von IBM ebenso erfolgreich durchgeführt: Die Konstruktion und Vermarktung von aufwärtsverträglichen Geräten, die dem Kunden nicht zumuten, gleich alle Software in den Müll zu werfen, sondern sie entweder über einfache Cross-Compiler oder aber auch unverändert mit hinüberzuziehen. Beim Prozessor des ATs gibt es da einen Betriebsmodus, der den 8088-/8086-Betrieb vollständig emuliert. Im AT4SLC selbst sind zwei Steckplätze für PC-Steck-Karten vorhanden, die vom Prozessor genauso bedient werden, als wären sie in einem PC eingebaut. Darüber hinaus gibt es sechs Steckplätze, die alle zusätzlichen 80286-Signale führen, die für den Speicherausbau auf 16 MByte notwendig sind. Eine interessante Feinheit ist, daß die Kompatibilität der Steckplätze AT versus PC doch nicht so 100 % beträgt, daß man sagen kann, alles läuft. Wie so oft im Einsatz der Computer muß man probieren. Außerdem haben die Programmierer bei IBM viele von Intel für weitere Entwicklungen reservierte Interrupts des 8088 doch im PC-System verwendet, so daß jetzt einige geheimnisvolle Kunstgriffe beim Betrieb des 80286 im AT notwendig sind. Zum Beispiel wird der Interrupt 0D bei IBM für den Drucker 2 genutzt, während der 80286 damit auf einen Segmentüberlauf reagieren will. An der Stelle kann man nur sagen, daß der AT soweit als möglich kompatibel zu den kleineren Modellen ist, nicht alles aber problemlos laufen wird. Keine Probleme wird der bekommen, der eine reine AT-Umgebung hat.

Ein bißchen Erfahrung

Während des Testes habe ich den Rechner von MCI auch aufgeschraubt und zum Beispiel nachgesehen, welches Diskettenlaufwerk eingebaut ist. Das Fabrikat stammt aus Japan von JVC. Mir scheint, als hätte der AT4SLC nicht nur billigste Komponenten eingebaut, die Speicherchips zum Beispiel stammten bei dem Modell im Test von NEC. Damit hat sich bei mir nicht bewahrheitet, daß möglicherweise RAM-Ausfälle nach kurzer Betriebszeit zu erwarten wären, weil die Chips bei preiswerten Angeboten miserabel sein sollen. Der Computer besitzt einen 16-MHz-Quarz, die CPU läuft also mit 8 MHz. Das Netzteil scheint von der erforderlichen Leistungsfähigkeit zu sein, es zieht etwa 2 A bei 220 V, das sind immerhin 440 W. Bei der Arbeit mit Euroscript habe ich den von dort vorgesehenen Warm-Boot-Ausstieg ins DOS durchgeführt. Das bedeutet, daß er mit der derzeit modernsten Betriebssystemversion alles an Segmentreservierung mitmacht. Allerdings soll DOS 3.2 auch nicht fehlerfrei sein. Das obengenannte Buch meint zum Beispiel, daß der Betrieb der Festplatte nicht nur wegen der Head-Crash-Gefahr erst mit systematischer Back-Up-Strategie sicher sei, sondern auch, weil die Routine im DOS, die die immer auf einer Platte vorhandenen Fehl-Sektoren bewacht, nicht zuverlässig arbeite und daher manchmal die Daten in einem „schwarzen Loch" verschwinden würden. Die Tastatur des Testrechners gefiel mir nicht sehr. Bei meiner Schreibweise konnte ich kein klares Gefühl dafür entwickeln, wann ein Anschlag gelungen war, wann nicht. Besonders das R habe ich oft doppelt angeschlagen. Mein Test mußte sich mangels geeigneter IBM-Adapterkarten und mangels Software auf den vielfältigen Gebrauch der Festplatte beschränken. Von dort lädt sich Turbo-Pascal schnell hoch, nur die vom System her definierbare RAM-Disk ist da noch schneller. 20 MByte sind erfreulich viel Platz.
Die herkuleskompatible Bildschirmkarte hat mit dem Monitor DM 14 zusammen sehr ordentliche Bilder geliefert.

Herwig Feichtinger

Janusкopf

Apples 65816-Computer II GS

Apple-intern wurde er zunächst unter dem Code-Namen „Cortland" geführt, jetzt nannten ihn die Marktstrategen „Apple-II GS". Apples 65816-Maschine blickt wie ein Janusкopf in zwei Richtungen, nämlich in die gute, alte Welt des Apple-II, andererseits aber auch in die Macintosh-Welt mit Maus- und Fenstertechnik.

Ein Apple-II-kompatibler Macintosh? Oder ein Apple-II mit Macintosh-Features? Verglichen mit dem gegenüber beiden Welten neutralen Namen „Cortland" deutet „Apple-II GS" eine nähere Verwandschaft mit dem Apple-II an. Und das ist nicht unberechtigt, wenn auch die Rollkugel-Maus und entsprechende Software und sogar der Disketten-Auswurf per Mini-Motor im Laufwerk genauso funktionieren wie beim Macintosh.

Zuerst zum technischen Aufbau. Das Gehäuse bietet auch für lange Steckkarten ausreichend Platz, da keine Floppy-Laufwerke eingebaut sind. Es läßt sich leicht öffnen. Die Tastatur *(Bild 1)* ist trotz des abgesetzten Ziffernfeldes auffallend kompakt und eignet sich durch leichte Erhöhungen auf einigen Tasten besonders gut für „blindes" Schreiben. Der Schwarzweiß- oder RGB-Monitor paßt auf das Gehäuse der Grundeinheit, läßt sich allerdings nicht schwenken (was gerade bei der fehlenden Entspiegelung nachteilig sein kann). Die mitgelieferte Maus ist die vom Macintosh bekannte Rollkugel-Ausführung. Bevor wir uns dem Innenleben des „Cortland" zuwenden, zunächst aber noch etwas zur Vorgeschichte.

Wozniaks Traum-Computer

Steve Wozniak, der zusammen mit Steven Jobs die kalifornische Firma Apple gegründet hatte, war der eigentliche Vater der Apple-II-Familie; er war der Konstrukteur, während Jobs sich vorwiegend um das Verkaufen kümmerte (was bekanntlich nicht weniger wichtig ist).

Wozniak hatte das Monitor-Programm der ersten Apples selbst geschrieben – inklusive Mini-Assembler, Disassembler und I/O-Routinen; lediglich den Applesoft-Basic-Interpreter hatte Apple später von Microsoft erworben.

Wer den alten Apple-Monitor noch kannte (er war im II-Plus schon nicht mehr so vorhanden), weiß, daß er eine Unterprogramm-Sammlung namens Sweet-16 enthielt, die der 8-Bit-CPU 6502 die Simulation einer Pseudo-16-Bit-CPU ermöglichte. Wozniak war sich damals schon im klaren darüber, daß er eigentlich einen 16-Bit-Apple mit einem 6502-Nachfolge-Chip bauen wollte, sobald dieser Chip da wäre.

Jahrelang war unsicher, ob es überhaupt einen 16-Bit-Nachfolger der einstmals von MOS Technology entwickelten 6502-CPU geben würde. Sowohl Synertek als auch Rockwell hatten solche Pläne, die sie dann jedoch wieder aufgaben, nachdem Rockwell sich für den 68000 von Motorola entschieden hatte. Wozniak ließ aber nicht locker und führte selbst Gespräche mit mehreren Halbleiterfirmen, um seinen Traum zu verwirklichen.

Ein Neuling in der CPU-Branche, GTE Microcircuits, ließ dann 1985 vom Western Design Center nach einigen Vorarbeiten mit dem G65SC02, einer CMOS-Version des 6502, doch einen Nachfolge-Chip bauen; zwar (wie Intels 8088 oder Motorolas 68008) keinen echten 16-Bitter, sondern mit einem externen 8-Bit-Datenbus, aber mit interner 16-Bit-Verarbeitung. Wesentliches Merkmal dieser CPU ist, daß durch Setzen eines Flag-Bits im Statusregister auch die Vorläufer-CPU vollständig emuliert werden kann. Von den ersten drei Computern, die mit dem seit Ende 1985 erhältlichen 65816 realisiert wurden, war keiner von Apple: ein nur in England vermarkteter „Communicator" von Acorn, der mc-65816-Computer und der Space-65 von IBS.

Über die in CMOS-Technik realisierte CPU 65816 war in mc ja schon mehrfach die Rede, sogar einen Selbstbau-Computer auf der Basis dieses Prozessors haben wir vorgestellt. Es ist kein Geheimnis, daß die Resonanz auf dieses mc-Projekt

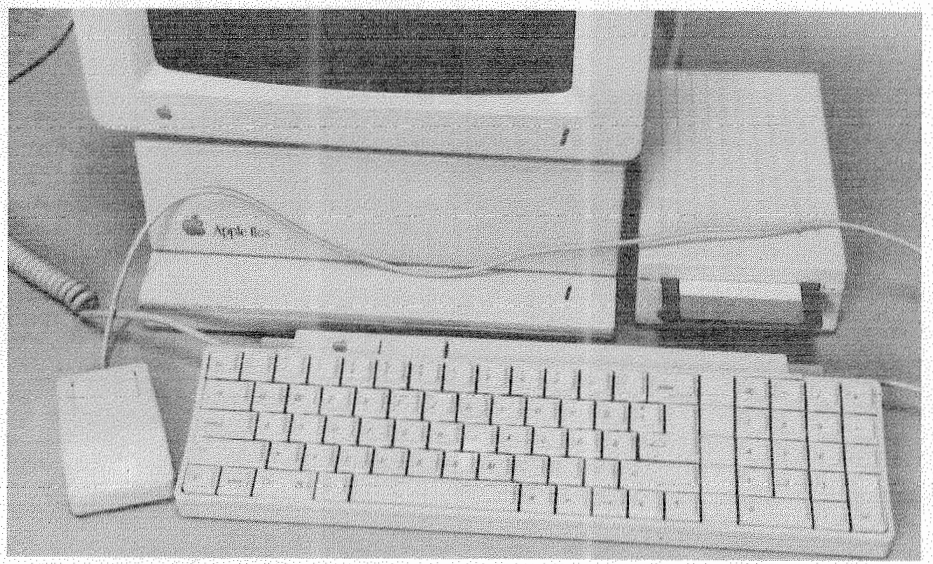


Bild 1. Die Tastatur des Apple-II GS nimmt nur wenig Platz auf dem Schreibtisch weg – die Maus dagegen um so mehr ...

geringer war als die z. B. auf den mc-68000-Computer; hoffentlich ist das kein böses Omen für Apple ...

Kundenspezifische Chips

Nachbauer aus Taiwan dürften es mit dem Apple-II GS schwer haben. Ein erheblicher Teil des Innenlebens besteht nämlich aus kundenspezifischen Chips. Den GS kann man sich als zwei Subsysteme auf einer Platine vorstellen (*Bild 2*). Auf der einen Seite ein Apple-IIe-ähnliches System mit 128 KByte RAM, der Video-Logik, internen und externen Slots und dem Mega-II-Chip, der nahezu die gesamte Logik eines herkömmlichen Apple-II auf einem IC realisiert. Auf der anderen Seite die neue 65816-CPU, das FPI-IC (Fast Processor Interface für Adressen-Zuweisung und Memory-Refresh) und, je nach Ausbau, bis zu 4 MByte schnelles dynamisches RAM.

Der interessanteste und komplexeste Chip ist wohl der „Mega-II". Er integriert die Adressendecodierung, einen Timing-Generator, die Video-Logik (im Zusammenwirken mit einem weiteren Video-Chip namens VGC), einen Zeichengenerator für acht nationale Zeichensätze, I/O-Schaltungen für Maus und Game-Port sowie die Refresh-Logik für das langsamere RAM. Sowohl die NTSC- als auch die PAL-Video-Norm werden unterstützt.

Im Normalfall läuft die CPU bei Zugriffen auf die Mega-II-Seite mit 1,023 MHz (wie ein „richtiger" Apple) und bei solchen auf die FPI-Seite mit 2,8 MHz. Dadurch ist ein korrektes Timing auch beim Zugriff auf herkömmliche Apple-II-Steckkarten gewährleistet.

Der Floppy-Controller heißt schlicht „IWM" (Integrated Woz Machine, frei nach Steve Wozniak) und ersetzt die bisherige Disk-Controller-Steckkarte. Das Funktionsprinzip GCR (Group Code Recording), von Wozniak von Anfang an verwendet, ist gleich geblieben: Jeweils sechs Datenbit werden trickreich so in ein 8-Bit-Muster konvertiert, daß beim Lesen von Diskette eine Taktrückgewinnung rein digital möglich ist. Eine Konvertierung von Apple-Disketten mit Computern, die herkömmliche Disk-Controller enthalten, ist deshalb nach wie vor aus Hardware-Gründen nicht möglich; ein Dateitransfer ist mit Fremdgeräten nur über die serielle Schnittstelle möglich. (Über dieses Problem ärgert sich die mc-Redaktion regelmäßig, wenn es um die Übernahme von Texten in den Satzrechner geht.)

Die Adressierung der Slots erfolgt über ein Custom-IC namens „Slotmaker"; er generiert auch die am Game-Connector zur Verfügung stehenden Ausgangssignale (Annunciators). Für die Abfrage der Tastatur über den „Front Desk Bus" ist ein eigener Mikrocontroller vorhanden.

Kompatibel mit den Vorgängern

Bedingt durch die Umschaltbarkeit der CPU in den 6502-Emulations-Modus kann der neue Apple die älteren Modelle der Apple-II-Familie praktisch vollständig emulieren. Die Taktfrequenz ist hierbei zwischen 1 MHz und 2,8 MHz umschaltbar, so daß viele Programme fast dreimal schneller als auf älteren Apple-Modellen laufen können. Im 2,8-MHz-Modus sind Applesoft-Basic-Programme immerhin etwa so schnell wie in GW-Basic auf einem IBM-PC-kompatiblen Computer (8088-CPU, 4,77 MHz).

Bei Video- und I/O-Zugriffen (Slots) wird jedoch grundsätzlich mit 1 MHz gearbeitet, um eine weitgehende Kompatibilität mit existenter Hardware und Software zu erreichen. Hundertprozentig ist das allerdings nicht gelungen: Bei Z80-Karten gibt es aufgrund dem beim Apple-II GS geänderten DMA-Timing Schwierigkeiten. Es wird jedoch sicher nicht lange dauern, bis passende Z80-Karten auf dem Markt sind.

Apple hat sich bei der Apple-II-Kompatibilität des GS große Mühe gegeben: Die intern schon vorhandene I/O-Hard- und Software liegt adressenmäßig so, wie es einer Standard-Slot-Konfiguration des Apple-IIe entspricht. Das gilt sogar für die Language-Karte des alten II-Plus, die man bei den standardmäßig vorhandenen 256 KByte RAM ja nicht mehr braucht. Trotzdem sind die sieben vorhandenen Slots im GS durchaus nutzbar, da sich die interne Slot-Belegung per Software einstellen läßt.

Speicheraufteilung

Im 6502-Modus sieht der Speicher prinzipiell so aus wie in einem Apple-IIe mit 128 KByte; die unteren 64 KByte bezeichnet Apple als „main memory", die oberen als „auxiliary memory". Sie entsprechen den Bänken 00 und 01 des 65816-Prozessors. Im 6502-Modus werden die Zero-Page und die Stack-Page,

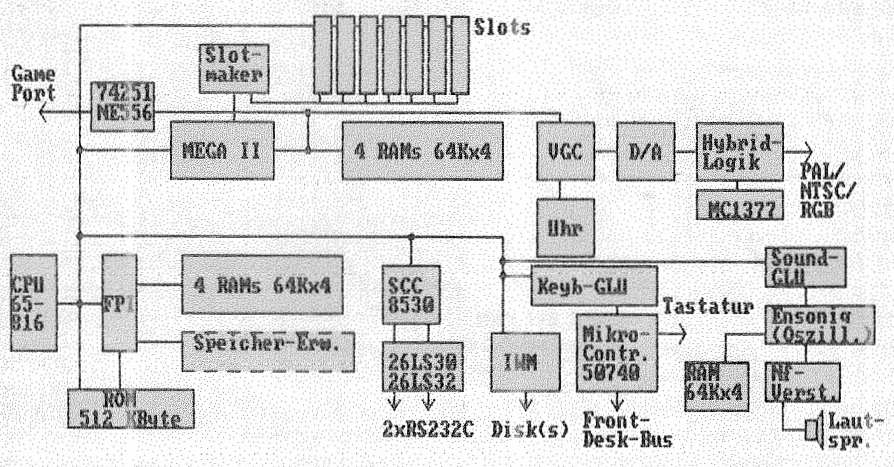


Bild 2. Blockschaltbild des „Cortland" von Apple. Der hochintegrierte Mega-II-Chip enthält einen großen Teil der Schaltung des Apple-IIe

Positives
- Schneller Prozessor
- Moderne Benutzeroberfläche
- Apple-II+/e/c-Programme lauffähig
- Komfortables Monitorprogramm im ROM
- Echtzeituhr und Sound-Chip
- Zwei serielle Schnittstellen
- 7 Slots, weitgehend Apple-II-kompatibel
- 800 KByte Kapazität bei 3,5-Zoll-Disks

Negatives
- Keine genormten Schnittstellenstecker
- Applesoft-Interpreter nicht verbessert
- Mitgelieferte Dokumentation unzureichend
- Bildschirm nicht entspiegelt
- Boot-Vorgang dauert lange
- Teurer als vergleichbare Systeme

d. h. die Adressen 0000 bis 01FF, immer in Bank 0 angesprochen, da sich der Prozessor sonst bei der Bank-Umschaltung „aufhängen" würde.

Maximal kann die CPU mit ihren 24 Adressenleitungen 16 MByte in Bänken von je 64 KByte adressieren. Bedingt durch die interne Adressendecodierung im „Cortland" können maximal 8 MByte RAM und 1 MByte ROM angesprochen werden. Speichererweiterungs-Karten sind derzeit mit 1 MByte (mit 256-KBit-Chips) lieferbar; eine Version mit 4 MByte Kapazität (mit dem neuen 1-MBit-Chips) ist geplant.

Außer dem normalen Arbeitsspeicher ist ein kleiner CMOS-Speicher vorhanden, der wie die Echtzeituhr von einer Batterie gepuffert wird (Batterie-Lebensdauer nach Angaben von Apple etwa acht Jahre) und wichtige Systemparameter wie nationaler Zeichensatz, Schnittstellen-Konfiguration usw. auch nach dem Ausschalten des Rechners speichert.

Die Schnittstellen

Im GS ist keine Disk-Controller-Steckkarte erforderlich. Der Anschluß für (externe) Diskettenlaufwerke ist als 19polige D-Sub-Buchse an der Rückseite herausgeführt. Sowohl 3,5-Zoll- (800 KByte Kapazität) als auch 5,25-Zoll-Laufwerke (143 KByte, Apple-II-kompatibel) lassen sich anschließen. Ein Kassettenrecorder-Anschluß ist übrigens nicht mehr vorgesehen.

Eine Kompatibilität zum „alten" Apple-II besteht auch bei den seriellen Schnittstellen. Sie sind von der Firmware her kompatibel mit der verbreiteten „Super Serial Card", allerdings nicht hardwaremäßig. Wenn ein Anwenderprogramm also unter Umgehung der Firmware direkt die Hardware adressiert, wird es nicht funktionieren. Ein simpler Terminal-Emulator z. B. zur Kommunikation mit einem Modem ist wie beim Apple-IIc aber bereits im ROM vorhanden.

Ein wunder Punkt ist, daß (wie bei Apple leider üblich) zwar zahlreiche Schnittstellen vorhanden sind, diese aber zumindest mechanisch keinerlei Normen entsprechen. So sind die beiden mit Zilog-SCC-Chips realisierten Schnittstellen zwar bis auf die fehlende RI-Leitung (Ring Indicator) elektrisch mit den Normen RS-232C und V.24 vergleichbar, nicht aber mechanisch, da zum Anschluß von Peripheriegeräten Spezialstecker erforderlich sind. Die einzigen halbwegs marktüblichen Anschlüsse sind der Composite-Video-Ausgang in Form einer Cinch-Buchse und der Audio-Ausgang, eine 9-mm-Stereo-Klinkenbuchse. Eine monophone Wiedergabe ist jedoch auch über den eingebauten Lautsprecher möglich.

Video-Ausgabe

Die Video-Ausgabe erfolgt wahlweise mit 50 oder 60 Hz Ablenkfrequenz ohne Halbbildverfahren, d. h. rund 300 Rasterzeilen. Da man am oberen und unteren Bildrand noch etwas Platz freilassen muß, ergibt sich automatisch eine Begrenzung der vertikalen Auflösung auf etwa 200 Punkte (übrigens wie bei der CGA-Karte, sprich Farbgrafik-Interface, bei IBM-PC-kompatiblen Computern). Wir testeten den Apple-II GS in einer Konfiguration mit dem dazugehörigen Farb-Monitor, der ein im Vergleich zu den alten Apple-Modellen gestochenes Farbbild liefert – nicht zuletzt deshalb, weil der Monitor über den RGB-Anschluß versorgt wird. Störend ist höchstens, daß die Bildröhre nicht entspiegelt ist, was man beim Preis des Monitors eigentlich erwarten dürfte.

Neben den schon bekannten Apple-II-Modi werden von der Grafik auch zwei neue Modi unterstützt, nämlich einer mit 320 × 200 Punkten bei 16 Farben (ein Pixel wird intern mit vier Bit gespeichert) und ein weiterer mit 640 × 200 Punkten bei 4 Farben (2 Bit je Pixel).

Die Software

Immerhin 128 KByte ROM enthält der Apple-II GS. Ihr Inhalt ist bis auf den Applesoft-Basic-Interpreter in den normalerweise mitgelieferten Handbüchern leider in keiner Weise dokumentiert. Dies ist um so bedauerlicher, als zum Beispiel das Monitorprogramm den 65816 voll unterstützt und ähnlich komfortabel wie jenes in den allerersten Apples ist (später fielen Teile davon der Disk-Boot-Routine zum Opfer): Disassembler, Mini-Assembler, Verschieben und Vergleichen von Speicherinhalten sind eine feine Sache für jeden Programmentwickler. Es ist mir rätselhaft, warum so etwas dem Käufer gegenüber schlicht verschwiegen wird. Wenn man im Benutzer-Handbuch unter „Monitor" nachschlägt, dann erhält man lediglich Auskunft über die lieferbaren Bildschirme...

Im ROM sind zahlreiche Routinen untergebracht, die dem Programmierer eine komfortable Ein- und Ausgabe, ggf. auch von der Maus unterstützt, ermöglichen: Diese „Toolbox" ist in ihren Funktionen mit jener des Macintosh vergleichbar, aber (wegen der anderen CPU) natürlich nicht code-kompatibel.

Auf Diskette mitgeliefert werden zwei Versionen des vom Apple-IIe und IIc bekannten Betriebssystems Prodos – eine 8-Bit-6502- und eine 16-Bit-65816-Version. Prodos/16 unterstützt hierarchische Subdirectories und den vollen Adressierungsbereich der 65816-CPU. Die jetzige Version ist allerdings nur eine Notlösung und wird Anfang 1987 durch Prodos/16(2) ersetzt werden.

Ein Selektor-Programm erlaubt das Aufrufen anderer Programme durch „Anklicken" des entsprechenden Symbols auf dem Bildschirm mit der Maus.

Ferner soll es auch eine Übungsdiskette geben, die dem Benutzer den Gebrauch von Rechner, Maus und Laufwerken vorführt; diese lag uns allerdings zum Test noch nicht vor.

Es ist schade, daß Apple den Applesoft-Interpreter nicht weiterentwickelt (oder diese Aufgabe an Microsoft übertragen) hat. Die primitiven Editiermöglichkeiten entsprechen nicht dem Stand der Technik; ein wirklicher Bildschirm-Editor war schon vor Jahren in Commodores PET-2001 eingebaut und ist heute bei Programmiersprachen für IBM-Kompatible selbstverständlich. Sogar das Prompt-Zeichen „Ü", das am Zeilenanfang statt der „eckigen Klammer zu" im deutschen Zeichensatz als Basic-Prompt erscheint, gibt es beim Apple-II GS im-

mer noch. Der Applesoft-Interpreter im ROM nutzt den erweiterten Befehlssatz und die Adressierungsmöglichkeiten der neuen CPU 65816 nicht und profitiert lediglich von deren höherer Taktfrequenz.

Es ist zu vermuten, daß ein erheblicher Teil der sonstigen System-Software von Apple nicht in Assembler, sondern in C geschrieben wurde, u. a., weil viele Systemaufrufe dem Macintosh sehr ähnlich sind. Leider dauert dadurch aber der Boot-Vorgang von der 3,5-Zoll-Systemdiskette beim Einschalten des Rechners mit Selektor und Mouse-Desk, also noch ohne irgend ein Anwenderprogramm, rund 50 Sekunden. Auch hier zeigt sich (leider) eine große Ähnlichkeit mit dem Macintosh. Eine 5,25-Zoll-Diskette läßt sich zum Booten praktisch nicht verwenden, weil ihre 143-KByte-Kapazität für die Systemdateien nicht ausreichen. (Es sei denn, man verwendet das alte DOS 3.3 oder reduziert die Zahl der Systemdateien auf ein Minimum, das für bestimmte einfache Anwenderprogramme gerade noch ausreicht.)

Zur Erstellung von Anwenderprogrammen können Softwarehäuser den „Cortland Programmer's Workshop" als Zubehör beziehen (Disk und Handbuch). Er enthält eine „Shell", die die Benutzeroberfläche mit Maus und Pulldown-Menüs realisiert, einen Editor zum Erstellen und Verändern von Quell-Files, einen Linker zum Erzeugen ausführbarer Programme, einen Debugger sowie einige Utility-Programme. Zusätzlich kann man, je nach bevorzugter Sprache, einen 65816-Assembler, einen C-Compiler oder einen Pascal-Compiler erwerben (letzterer ist übrigens nicht identisch mit dem bekannten UCSD-Pascal für den Apple-II).

Die Dokumentation

Zum Apple-II GS gehören vier Handbücher: ein Heft mit Installations-Anweisungen; das Apple-II-GS-Benutzerhandbuch mit allgemeinen Hinweisen über den Umgang mit dem Rechner; ein Finder-Handbuch, das uns zum Test leider nicht vorlag; und schließlich ein Handbuch, das den Basic-Interpreter halbwegs beschreibt. Halbwegs deshalb, weil eine Reihe von Befehlen nicht so beschrieben werden, daß man mit ihnen wirklich umgehen könnte: Weder enthält es Hinweise, auf welche Adressen man wofür PEEK, POKE und CALL anwenden kann, noch beschreibt es die Programmierung von Grafik-Shapes für den DRAW-Befehl.

Eine Reihe von technischen Handbüchern, die auf Anforderung kostenpflichtig erhältlich sein sollen, befand sich bei Redaktionsschluß in Vorbereitung; wir konnten einen Blick in fotokopierte Vorversionen tun, die im Prinzip genau das enthalten, was man sich von einer „richtigen" Dokumentation erwartet. Ob der Apple-II GS als „offener Computer" betrachtet werden kann, wird nicht zuletzt davon abhängen, an wen und zu welchem Preis diese für ernsthafte Anwender unverzichtbare Dokumentation abgegeben werden wird.

Zu spät? Zu teuer?

Ob sich ein Computer seinen Markt erobern wird, ist immer schwierig vorherzusagen. (Ich habe mich später zum Beispiel darüber gewundert, daß plötzlich alle Welt den IBM-PC nachbaute, obwohl er zum Zeitpunkt des Erscheinens eigentlich schon veraltet war.)

Nicht wenige meiner Bekannten, alles frühere Apple-Anhänger, haben sich inzwischen einen preiswerten Taiwan-PC besorgt. Trotzdem wird es noch viele Apple-II-Besitzer geben, die lieber auf ein Gerät umsteigen würden, bei dem sie nicht ihre gesamte Software auf den Müll werfen müßten. Gerade die Kompatibilität mit dem alten II-Plus führte aber natürlich auch zu Kompromissen.

Man könnte den neuen Apple mit dem Versuch vergleichen, eine Raumfähre zu bauen, in der man auch alte Doppeldekker-Pilotensitze verwenden kann.

Der Prozessor 65816 ist vorwiegend deshalb interessant, weil er den Vorläufer 6502 emulieren kann und dabei dreimal schneller ist. Vor allem wegen der Speicherorganisation in Bänken ist er für den Programmierer bei umfangreichen Daten ähnlich umständlich zu handhaben wie der vom IBM-PC bekannte 8088 von Intel und dem 68000 an Komfort deutlich unterlegen.

Eine große Chance hat der II GS dennoch: Wegen der großen Beliebtheit von Einplatinen-Computern (mc-EMUF, mc-65816-Computer u. v. a.) könnte Apples Wunderknabe ein ideales Entwicklungssystem für solche Geräte werden. Voraussetzung dafür ist allerdings, daß auch dem Endanwender eine möglichst vollständige Dokumentation zu vernünftigen Preisen zur Verfügung gestellt wird, was abzuwarten bleibt.

Der Apple-II GS kostet in einer anwendbaren Konfiguration immerhin rund 4000 DM. Zwar war darüber bei Redaktionsschluß das letzte Wort noch nicht gesprochen, und tatsächlich geliefert werden soll das Gerät an Endkunden nicht vor Februar. Man muß sich allerdings vor Augen halten, was man für 4000 DM auch bekommen könnte: zwei Atari-STs oder drei IBM-PC-Kompatible. Oder auch zwei Apple-II-kompatible 65816-Rechner (Space-86 von IBS).

Wenn man das Apple-II-GS-Gehäuse öffnet, sieht man auf einen Blick, daß wesentlich weniger drin ist als in einem Taiwan-PC – eine direkte Folge der weitgehenden Integration mit kundenspezifischen ICs wie Mega-II und IWM. Wenn deren Entwicklungskosten wieder hereingeholt sind, ist sicher noch Luft für Preissenkungen beim GS.

Also, Apple, nichts für ungut, aber: Entweder wird er preiswerter, oder das Schicksal von Wozniaks Traum-Computer ist jetzt schon besiegelt. Und das wäre wirklich schade, steckt doch eine Menge guter Ideen in diesem System.

Technische Daten des Apple-II GS

Prozessor: G65SC816 (GTE), 1 MHz oder 2,8 MHz

RAM: 256 KByte in 64-KByte-Bänken

ROM: Applesoft-Basic (wie IIe), Toolbox (für I/O-Aufrufe von Anwenderprogrammen), 65816-Monitor mit Disassembler und Miniassembler für 24-Bit-Adressen, Control-Panel für Modus-Umschaltung usw.

Betriebssysteme: Prodos 8, Prodos 16, DOS 3.3

Tastatur: 78 Tasten, abgesetztes Zifferfeld

Video-Ausgang: RGB oder (F)BAS

Textdarstellung: 40/80 Zeichen pro Zeile

Grafikauflösung: max. 640 × 200 mit 4 Farben; Apple-IIe-Grafik kann emuliert werden

Schnittstellen: 7 Slots, 2 × V.24 (für Drukker, Appletalk-Netzwerk, Modem usw.), Disk-Port (IWM), Game-Port, Desktop-Bus (für Tastatur und Maus)

Disk: 3,5 Zoll (800 KByte) oder 5,25 Zoll (143 KByte), extern

Sound: Synthesizer-IC mit 32 Oszillatoren

Sonstiges: Tastaturpuffer, Echtzeituhr

Dieter Strauß

Harddisk auf Steckkarte

Megacard 20 und Megacard 32 im Test

Festplatten, die zusammen mit der Controllerkarte eine Einheit zum Einstecken in den PC bilden, kommen von zunehmend mehr Anbietern auf dem Markt. mc hat sich daher zwei dieser Erweiterungskarten, die Megacard 20 und die Megacard 32 von Merlin Elektronik, näher angesehen.

Viele PC-Benutzer haben sich aus Kostengründen einen PC ohne Festplatte gekauft. Bei intensiver Benutzung des PCs stellt sich bald heraus, daß der Betrieb mit Disketten auf die Dauer sehr unkomfortabel ist. Mit einer Festplatte hat man seine Programme und Daten viel schneller „im Kasten". Zum Aufrüsten des PCs mit einer oder gar zwei Festplatten gibt es mehrere Möglichkeiten: Tausch eines Floppy-Laufwerks mit einer Festplatte und einer separaten Controllerkarte, externe Plattensubsysteme und Kombination Festplatte/Controller als Steckkarte.

Die letztgenannte Lösung hat den Vorteil, daß zum Einbau nur der Gehäusedeckel des PCs entfernt und die Karte eingesteckt werden muß. mc hat die Megacard 20 und die Megacard 32 (*Bild 1*) von Merlin Elektronik in einen Commodore PC 10 eingebaut und getestet.

Beide Karten werden in einer gegen Stöße gut gesicherten Verpackung und einer Anleitung zur Inbetriebnahme geliefert. Die Megacard 20 besteht aus der 3½-Zoll-Festplatte D3126 von NEC und dem Controller WD 1002-WX von Western Digital. Die Megacard 32 besteht aus der gleichen Festplatte – sie ist jedoch mit dem Controller WD 1002-27X von Western Digital bestückt. Eine Megacard belegt eineinhalb Erweiterungplätze im PC. Es kann noch eine kurze Steckkarte, die bis maximal 160 mm lang sein darf, in den zur Hälfte von der Megacard benutzten Erweiterungssteckplatz eingesetzt werden. Die technischen Daten der beiden Karten sind in der *Tabelle* kurz zusammengefaßt.
In der mitgelieferten Einbauanleitung wird die Installation der beiden Karten ausführlich für den IBM PC, den Commodore PC und den Olivetti M 24 beschrieben. Beim Commodore PC ist unter dem Abschnitt „Gehäuse-Deckel abnehmen" sogar vermerkt, daß „5 Schrauben hinten und jeweils eine Schraube vorne an den beiden Seiten " entfernt werden muß. Mit dieser detaillierten Beschreibung dürften sogar der Mechanik völlig unkundige Benutzer zurechtkommen. Die Megacard 20 kann als erstes Laufwerk oder als zweites Laufwerk konfiguriert werden. Wird sie als erstes Laufwerk (Portadresse 320H, BIOS-Adresse C8000H) verwendet, so braucht an den werksseitig eingestellten Steckbrücken nichts geändert werden. Soll sie als zweites Laufwerk (Portadresse 324H, BIOS-Adresse CA000H) eingesetzt werden, so müssen lediglich zwei Steckbrücken geändert werden.

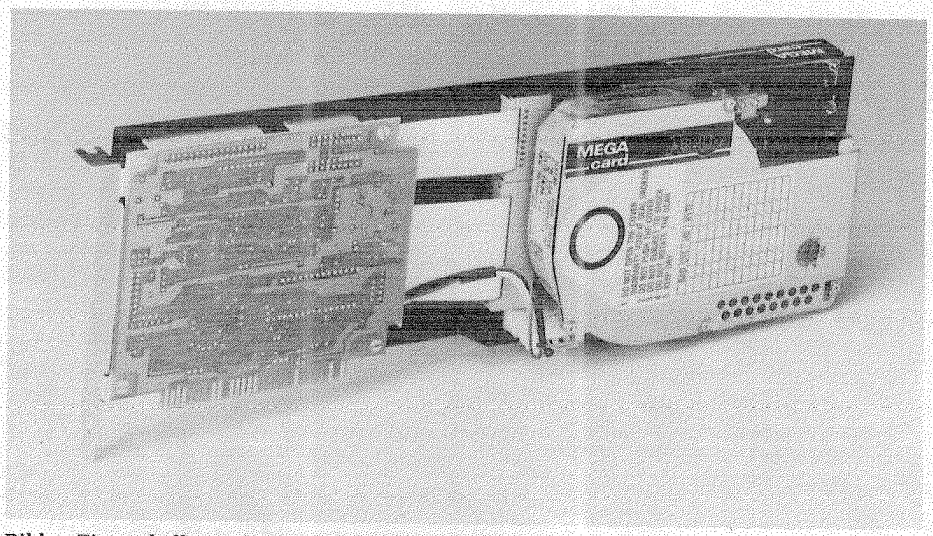


Bild 1. Eine geballte Ladung von 32 MByte auf einer Steckkarte

Einfache Inbetriebnahme

Die Megacard wiegt 1,27 kg und sollte daher mit beiden Händen vorsichtig in den Steckplatz eingepaßt und in einen der Stecker der Systemleitplatte eingedrückt werden. Die Karte muß wegen ihres Gewichts gut mit dem Befestigungsbügel angeschraubt werden. Nach dem Einbau muß der Abstand zu anderen Steckkarten geprüft werden. Überstehende lange Anschlußdrähte von benachbarten Platinen könnten an der Metalloberfläche der Platte einen Kurzschluß erzeugen. Sind die benachbarten Karten Qualitätsprodukte, dürfte dieses Problem nicht auftauchen. Beide Karten stoßen genau an der Kante der nächsten Erweiterungssteckleiste an. Es war aber dennoch möglich eine weitere Karte hier einzustecken. Während des Tests wurde die Videokarte unmittelbar neben der Festplatte gesteckt, um mögliche thermische Probleme erkennen zu können.
Der Gehäuse-Deckel kann nach dem Einbau der Karte wieder geschlossen, der Netzstecker eingesteckt und das System eingeschalten werden. Das System wird nach dem Einschalten die BIOS-Erweiterung des Festplatten-Controllers erkennen und booten. Das ganze wird jetzt etwas länger dauern, da erst die Befehle der BIOS-Erweiterung durchgeführt werden müssen. Nachdem MS-DOS geladen ist, kann die Festplatte physikalisch formatiert werden. Dazu muß DEBUG aufgerufen werden. *Bild 2* zeigt wie die physikalische Formatierung der Megacard 20 durchgeführt wird. Bei der Megacard 32 muß ein größerer Versatz (Interleave-Factor) gewählt werden, da die Megacard 32 wegen der RLL-Codierung (RLL = Run Length Limited) die Daten schneller überträgt als sie der Standard-PC verarbeiten kann. Nach der Meldung des Debuggers mit AX 0000:

muß bei dieser Karte vier eingeben werden. Es dauert nun einige Minuten bis die Platte physikalisch formatiert ist und sich das System mit „Format Successful" meldet. Mit FDISK wird die Platte partitioniert und danach mit Format C: logisch formatiert. Beim Commodore FDISK-Programm ist zu beachten, daß die neu erstellte DOS-Partition unter einem anderen Menüpunkt aktiviert werden muß. Mit SYS C: werden die System-Dateien und mit COPY a:*.* c: die restlichen Dateien der System-Diskette übertragen. Danach läßt sich das System nach dem Einschalten von der Platte booten.

Der Redaktion wurde während dieser Tests eine kleine Lektion zum Thema speicherresidente Programme erteilt. In der AUTOEXEC.BAT-Datei der Systemdiskette hatten wir ein speicherresidentes Programm zum Anpassen der verschiedenen Druckformate unseres Drukkers aufgerufen. Während der Initialisierungsphase der Megacard 20 gab es damit keine Probleme. Bei der Megacard 32 wurde die logische Formatierung mit der Fehlermeldung „Spur 0 fehlerhaft – Medium nicht verwendbar" abgebrochen. Nachdem aus der AUTOEXEC.BAT-Datei der Aufruf des speicherresidenten Programms entfernt war, ließ sich die Platte einwandfrei formatieren. Dieses Vorkommnis zeigt, daß sich in der Vergangenheit immer „brav" verhaltene speicherresidente Programme sich völlig unerwartet störend bemerkbar machen können.

```
A>debug
-r ax
AX 0000
:3
-g=c800:5
WX2 Format Revision 7.0 (C) Copyright Western Digital Corp. 1985
   (AH) = Relative drive number (0 - 7)
   (AL) = Interleave factor (3 is standard)
Press "y" to begin formatting drive C with interleave 03
```

Bild 2. So wird die Karte initialisiert

Die beiden Karten haben sich gegenüber weiteren Festplatten-Controllern im System völlig unterschiedlich verhalten. Einen zweiten Festplatten-Controller verträgt die Megacard 20 nur, wenn dieser auf Portadresse 324H und BIOS-Adresse CA000H konfiguriert ist. Die Megacard 32 hingegen verträgt keinen zweiten im System. Dafür kann der Controller der Megacard 32 eine zweite Festplatte ansteuerun. Dies ist einem Anfänger jedoch nicht zu empfehlen, da in diesem Fall Kabelsätze angefertigt werden müssen.

Thermische Probleme?

Bei der Diskussion für und wider Festplatten auf Einsteckkarten taucht immer wieder die Frage nach der thermischen Belastung der Festplatte auf. Megacard 20 und Megacard 32 nehmen jeweils 19 W auf. Diese Leistung wird während des Betriebs zum größten Teil in Wärme umgewandelt. Die gesamte Rückseite der Einsteckkarte ist daher aus Blech um die Wärme gut abzuleiten. Wir haben uns während des Betriebs überzeugt, ob die Wärme tatsächlich gut abgeleitet wird. Dazu haben wir an verschiedenen Stellen der Festplatte, der Rückseite der Einsteckkarte und auf dem Netzteil des PCs Temperaturmeßstreifen angebracht. Mit den Temperaturmeßstreifen von Kager lassen sich mit einem Streifen acht verschiedenene Temperaturen nachweisen. Der von uns verwendete Typ spricht an bei 40,6, 43,3, 46, 49, 54,4 60, 65,5 und 71 °C. Wird im Betrieb eine dieser Temperaturen erreicht, so verfärbt sich der jeweilige Punkt dauerhaft. Mit dieser Methode können Temperaturen nachgewiesen werden, ohne daß man im PC elektronische Temperatursensoren anbringen und einen Kabelbaum zu den Sensoren legen muß.

Nachdem die Temperaturmeßstreifen aufgebracht waren, haben wir den PC drei Stunden lang betrieben und anschließend den Gehäusedeckel wieder geöffnet. Kein Punkt außer einem an der seitlichen Kante der Festplatte hatte sich verfärbt! An der seitlichen Kante der Festplatte zeigte der Streifen 46 °C an. Die Festplatte hat sich somit trotz der geringen Abstände nicht übermäßig erhitzt. Zu diesem positiven Ergebnis trägt wesentlich die massive Rückwand der Festplattensteckkarte bei. Mit der Hand konnte deutlich gefühlt werden, daß die gesamte Oberfläche der Rückwand gleichmäßig erwärmt war. Die schwarze Oberfläche der Karte dürfte weiter dazu beitragen die Verlustwärme gut abzustrahlen. Am 120-W-Netzteil des PCs war keine Erwärmung festzustellen.

Mit der Megacard 20/32 dürften in einem Standard-PC keine thermischen Probleme auftreten, wenn nicht der PC bereits durch andere Zusatzkarten thermisch „ausgereizt" ist.

Zusammenfassend kann gesagt werden, daß Megacard 20 und Megacard 32 aus qualitativ hochwertigen Teilen aufgebaut sind, die sich in unseren Tests als zuverlässig erwiesen haben.

Tabelle: Die technischen Daten der beiden Karten

	Megacard 20	Megacard 32
formatierte Speicherkapazität	20 MByte	32 MByte
Platten	2	2
Köpfe	4	4
Zylinder	615	615
Aufzeichnungsverfahren	MFM	RLL
minimale Zugriffszeit	18 ms	18 ms
durchschnittliche Zugriffszeit	85 ms	85 ms
maximale Zugriffszeit	215 ms	215 ms
Startzeit	max. 15 s	max. 15 s
Stoppzeit	max. 15 s	max. 15 s
Zuverlässigkeit (MTBF)	20 000 h	20 000 h
Leistungsaufnahme	max. 19 W	max. 19 W
Umgebungstemperatur	+5 ... +45 °C	+5 ...+45 °C
Gewicht	1,27 kg	1,27 kg
Abmessungen:		
Länge	339 mm	339 mm
Breite	105,5 mm	105,5 mm
Höhe	42 mm	42 mm

Erhard Scherer

Der EMUF08

Ein starker Einplatinencomputer

Seit 1981, als der erste EMUF, eine Einfach-Euro-Karte mit 6504 CPU, von Herwig Feichtinger vorgestellt wurde, hat das Einplatinen-Computer-Konzept tausendfach Verwendung im privaten, industriellen und universitären Bereich gefunden. Seit der Zeit des 6504-EMUF ist die technische Entwicklung im Bereich der Mikroelektronik mit atemberaubender Geschwindigkeit vorwärts gestürmt. Die Integrationsdichte der Chips stieg um ein Vielfaches, für Speichergrößen, die noch vor einigen Jahren ganze Karten belegten, genügt heute ein einziger Baustein. Parallel zur Steigerung der Integrationsdichte ist ein deutlicher Preisverfall zu beobachten, der einen 68008-EMUF erschwinglich macht.

Heute kostet beispielsweise eine 68000-CPU (16 Bit) etwa genauso viel, wie vor wenigen Jahren ein Z80 (8 Bit). Das Aufkommen der 16-Bit-CPUs brachte Architekturen in den Mikrocomputerbereich, wie sie vorher nur bei sehr teuren Minicomputern, die zudem ganze Schaltschränke füllten, anzutreffen waren. Die 68000er Familie ist dafür ein sehr gutes Beispiel. Kenner der PDP11 entdecken sehr viele Parallelen zwischen beiden Rechnertypen, bezüglich des Registersatzes, der Maschinenbefehle usw.

Der neue EMUF vereinigt nun die Preiswürdigkeit der 8-Bit-EMUFs mit Architekturmerkmalen von Minicomputern. Das Herz des EMUF08 ist ein 68008. Dank des 8 Bit breiten äußeren Datenbusses (intern im Chip selbst sind die Busse 16 Bit breit) ist der 68008 im platzsparenden 48-Pin-Gehäuse untergebracht. Die 680xx-Familie hat sich inzwischen einen festen Platz als Leistungsträger in industriellen Anwendungen erobert; aber auch in Personal Computern wie dem Macintosh, dem Atari 520 oder dem Amiga gibt ein 68000er den Ton an. Die Verbreitung und Bekanntheit des 68000 läßt den Wunsch nach einem preiswerten Einplatinencomputer mit 68000-CPU für Steuer-, Meß- und Regelaufgaben aufkommen. Der EMUF08 erfüllt diesen Wunsch.

Die extreme Registerstruktur sowie der leistungsfähige Maschinenbefehlssatz des 68000 machen diesen EMUF für Aufgaben fit, bei denen Höchstleistungen gefordert sind. Der 68008 besitzt 8 Datenregister mit 32 Bit Breite (dies führt übrigens zu der Hochstapelei mancher Platinenhersteller, die Systeme mit dem 68000 als 32-Bit-Systeme verkaufen) und 8 Adreßregister (32 Bit breit), wobei A7 der Stackpointer ist. Es gibt zwei Betriebsmodi für die CPU: den User Mode, der mit einigen Einschränkungen verbunden ist, und den Supervisor Mode. Für beide Modi kann ein eigener Stackpointer (A7' bzw. A7) eingerichtet werden. Programme für den EMUF werden vorzugsweise im Supervisor Mode gefahren, da die Benutzung des User Mode nur in Multitasking/Multiuser-Systemen sinnvoll ist und zudem von der Hardware unterstützt werden sollte.

EMUF08 und Software

Die Erstellung von Software, in Assembler oder einer höheren Programmiersprache (z. B. „C") für dem EMUF kann auf den eingangs erwähnten Maschinen vom Apple, Atari, Commodore erfolgen – oder natürlich auch auf dem mc-68000-Computer, dem NDR-Klein-Computer, Hermann-20 von MTC Berlin (eine echte 32-Bit-Maschine mit 68020 CPU) oder dem c't-68000. Der Vorteil, sowohl auf dem Entwicklungssystem als auch auf der Zielhardware (in unserem Fall der EMUF) eine CPU vom selben Typ zu haben, liegt darin, daß die Assemblerprogrammierung nur einmal erlernt werden und auch während der Arbeit nicht ständig umgedacht werden muß. Ein weiterer Gesichtspunkt ist, daß Compiler, Assembler usw. nur einmal gekauft werden müssen. Wer einmal in der Assemblerprogrammierung den Schritt von den guten alten 8-Bit-CPUs zum 68000 getan hat, ist so verwöhnt, daß ihn ein Z80, 8085 oder 6800 nicht mehr reizen können. Ganzzahlarithmetik, mit 32 Bit breiten Zahlen, gibt's beim EMUF08 serienmäßig eingebaut und ohne Aufpreis. Die 68008-CPU besitzt Addition, Subtraktion, Multiplikation (16×16 Bit, 32 Bit Ergebnis) und Division (32/16 Bit, 16 Bit Ergebnis, 16 Bit Rest) bereits als Maschinenbefehle.

Beispiel: DIVS D1,D0

Das ist die vorzeichenbehaftete Division, mit dem Ergebnis in den niederwertigen 16 Bit von D0, dem Rest in den höherwertigen 16 Bit von D0. Diese Fähigkeit zum Jonglieren mit Ganzzahlen eröffnet dem EMUF08 Anwendungen, bei denen 8-Bit-CPUs aus Geschwindigkeitsgründen passen müssen, denn Integerarithmetik in Software ist langsamer als es die gewissermaßen in Hardware gegossenen Mikroprogramme des 68000 sind. Wer von den algorithmischen Höhen nicht ganz bis auf den Grund der Assemblerprogrammierung herabsteigen möchte, kann natürlich auch mit einer höheren Programmiersprache arbeiten. Die im Vergleich zu 8-Bit-CPUs größere Leistungsfähigkeit erlaubt es, mit dem EMUF08 Aufgaben in einer höheren Programmiersprache zu lösen, für die sonst aus Geschwindigkeitsgründen nur Assembler in Frage käme. Programme in höheren Programmiersprachen benötigen nicht nur mehr Platz als gleichwertige Assemblerprogramme, sondern sind in ihrer Ausführung auch langsamer. Höhere Programmiersprachen haben natürlich auch Vorteile: der Quellcode ist kürzer als in Assembler, die Programme sind im allgemeinen lesbarer und somit leichter zu pflegen. Nicht zuletzt sei die leichtere Übertragbarkeit auf eine andere Hardware erwähnt, für die z. B. „C" bekannt ist.

Die Hardware des EMUF

Der EMUF08 ist auf einer Einfach-Euro-Karte (100× 160 mm) untergebracht, ein Viertel der Platine ist Lochrasterfeld, drei Viertel belegt die Elektronik. Sie besteht (*Bild 1*) aus der 68008-CPU, RAM (8 KByte bis 32 KByte), EPROM (8 KByte bis 64 KByte), einigen TTL Bausteinen, sowie wahlweise einer VIA

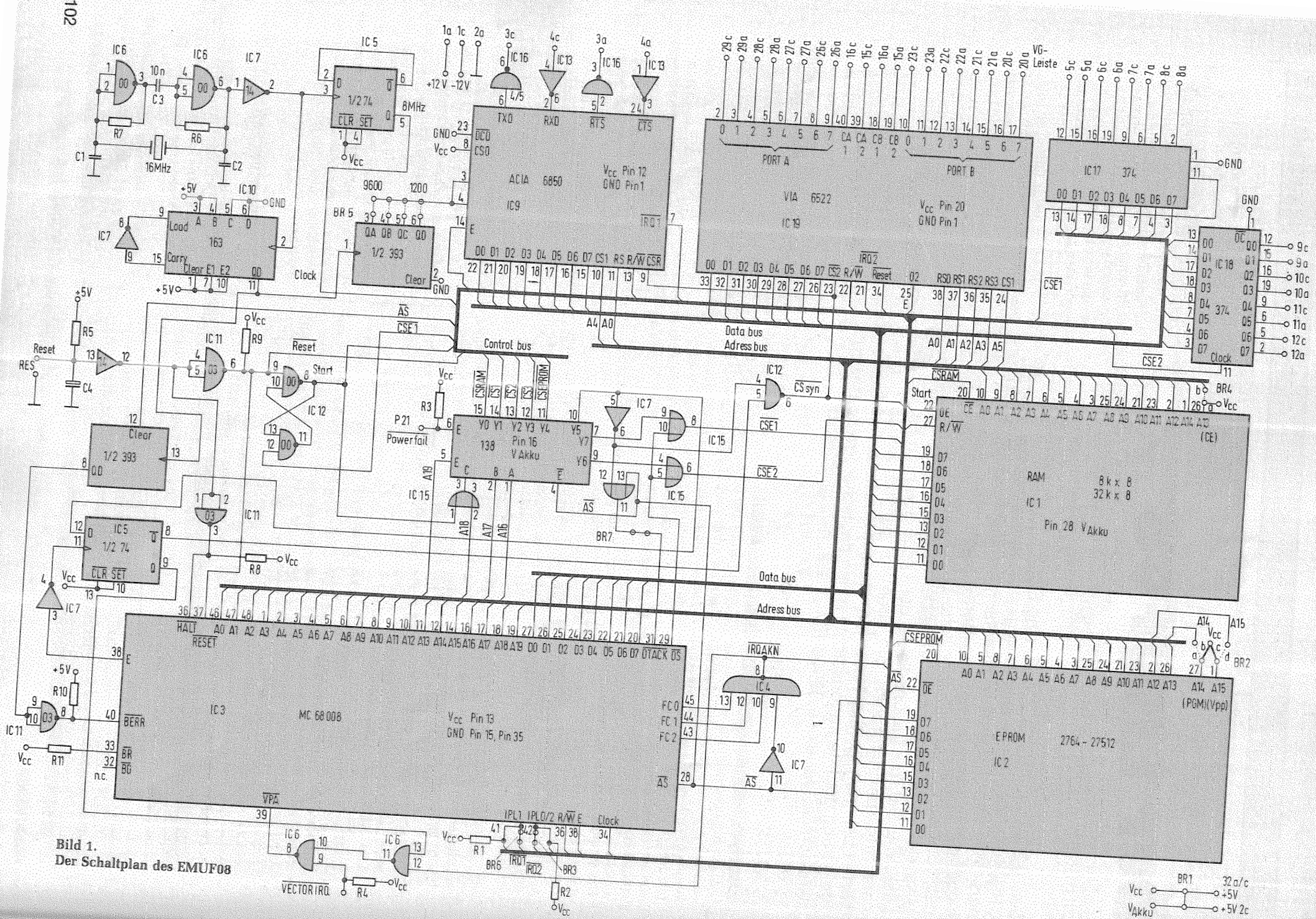


Bild 1.
Der Schaltplan des EMUF08

(6522), einer ACIA (6850) und zwei 8-Bit-Latches als Ausgängen. Damit stehen 18 Ein-/Ausgabe-, 2 Eingabe- und 16 Ausgabeleitungen zur Verfügung, insgesamt 36 parallele Leitungen. Die 16 Ausgänge der beiden Latches können, bei Verwendung von 74AS374 Bausteinen, 64 mA (bei 0-Pegel) Last treiben. Relais, LED Anzeigen usw. kann man also direkt anschließen. Die parallelen I/O Leitungen, die serielle Schnittstelle und die Stromversorgung führen auf eine 64polige VG-Leiste. Der „Anbau" von Zusatzhardware wie Anzeigen, Analog-Digital-Wandler, Relais ... ist über diesen Stekker durchzuführen. Die wichtigsten Anschlüsse der CPU und einige andere Signale führen auf eine Doppelreihe des Lochrasterfeldes (*Bild 2* zeigt die Lötseite der Platine, *Bild 3* die Bestückungsseite und *Bild 4* den Bestückungsplan). Wer mag, kann zum Beispiel über eine Pfostensteckverbindung Hardware aus eigener Entwicklung anschließen. Dafür steht ihm der Adreßraum ab hexadezimal 80000 bis FFFFF zur Verfügung. Dies sind 512 KByte.

Hinweise für den Aufbau

Für das Löten und Bestücken hat sich folgende Vorgehensweise als zweckmäßig erwiesen: Zuerst werden die Bauteile mit der niedrigsten Bauhöhe eingelötet, dann die mit der nächsthöheren, usw. Da immer Bauteile mit derselben Bauhöhe zu löten sind, können die Bauteile mit einer starken Pappe oder etwas ähnlichem gegen Herausfallen gesichert werden. Dieser Sandwich wird dann um 180 Grad gedreht, und die Bauteileanschlüsse können verlötet werden. Dabei stören die im vorherigen Arbeitsgang eingelöteten Bauteile nicht, da sie ja schon fest mit der Platine verbunden sind und zudem eine geringere Bauhöhe haben. Die „teuren" Bausteine, wie z. B. den MC68008, lötet man am besten nicht direkt ein, sondern stekt sie auf einen Sockel. Das Sockeln ist auch bei RAM und EPROM zweckmäßig, beim EPROM um das Austauschen bei einer Softwareänderung einfach zu gestalten, beim RAM um den Wechsel von den 8-KByte-RAMs auf die 32-KByte-RAMs ohne Lötkolben durchführen zu können. *Tabelle 1* zeigt die Stückliste.

Der EMUF08 muß, um funktionsfähig zu sein, nicht unbedingt voll bestückt werden. Folgende Bausteine können erst einmal weggelassen werden: RS232-Treiber (IC13, IC16), ACIA 6850, VIA 6522 und die Latches 74AS374. CPU, EPROM, RAM und Decodierlogik reichen für erste Funktionstests aus.

Der Anschluß der Versorgungspannung erfolgt über die 64polige VG-Leiste (siehe *Tabelle 2*, Steckerbelegung). Über diese Leiste können auch anwenderspezifische Schaltungen an den EMUF angesteckt werden, dies werden in der Regel Schaltungen sein, die VIA, ACIA oder die Latches benutzen. Der EMUF08 benötigt 5 V Versorgungsspannung, das Netzteil sollte für 1 A gut sein, um etwas Reserve für Zusatzschaltungen zu haben. Bei Benutzung der ACIA und der RS232-Treiber werden noch +12 V und −12 V benötigt (je ca. 50mA).

Die Anpassung an Bestückungsvarianten

Die Platine des EMUF08 ist mit einer Reihe von Steckbrücken ausgestattet, die eine einfache Anpassung an unterschiedliche Bestückungsvarianten und Betriebsarten ermöglichen. Bevor die Versorgungsspannung angelegt wird, sollte eine allgemeine Sichtkontrolle auf Löt- und Bestückungsfehler sowie auf die richtige Lage der Steckbrücken erfolgen (*Tabelle 3*).
Ein erster Funktionstest des EMUF kann mit einem einfachen Terminal (auch mit Hostcomputer) mit Hilfe von MONI-E8 (im EPROM) erfolgen. Falls beides nicht zur Verfügung steht, kann das untenste-

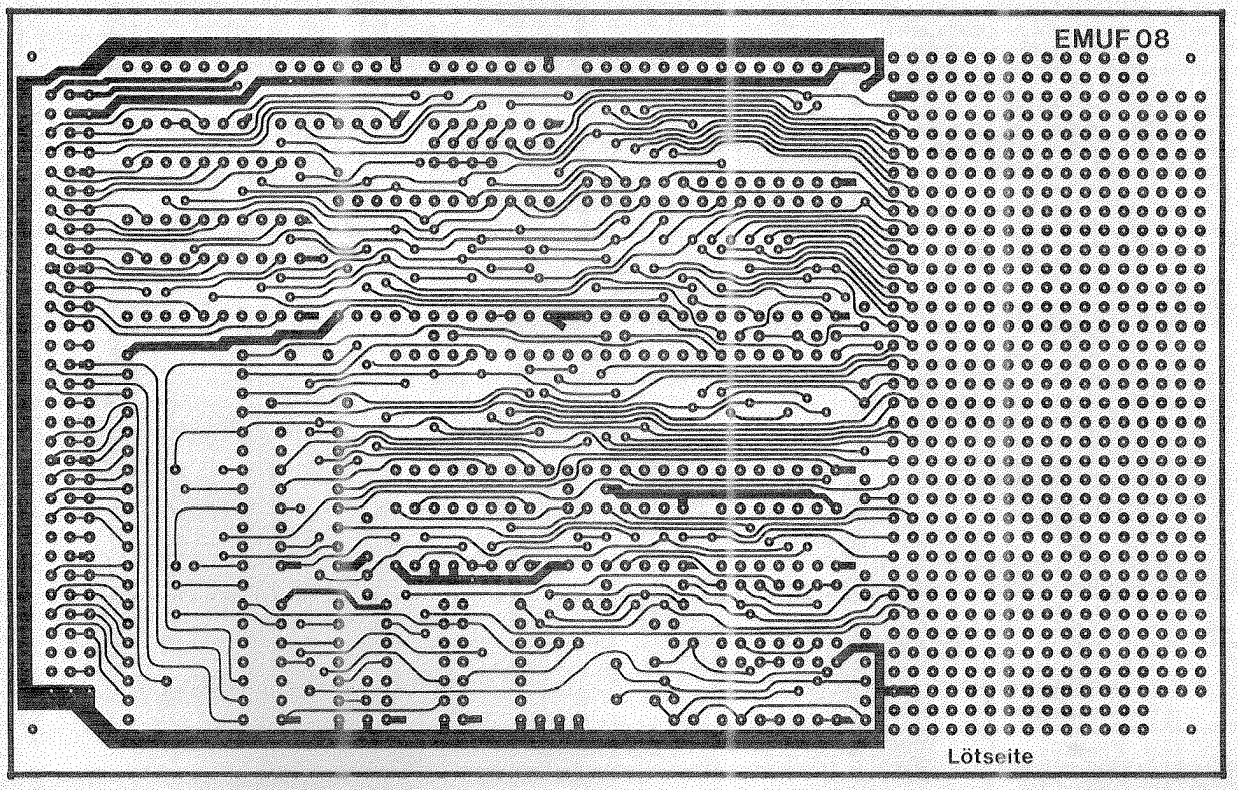


Bild 2. Die Lötseite der Platine

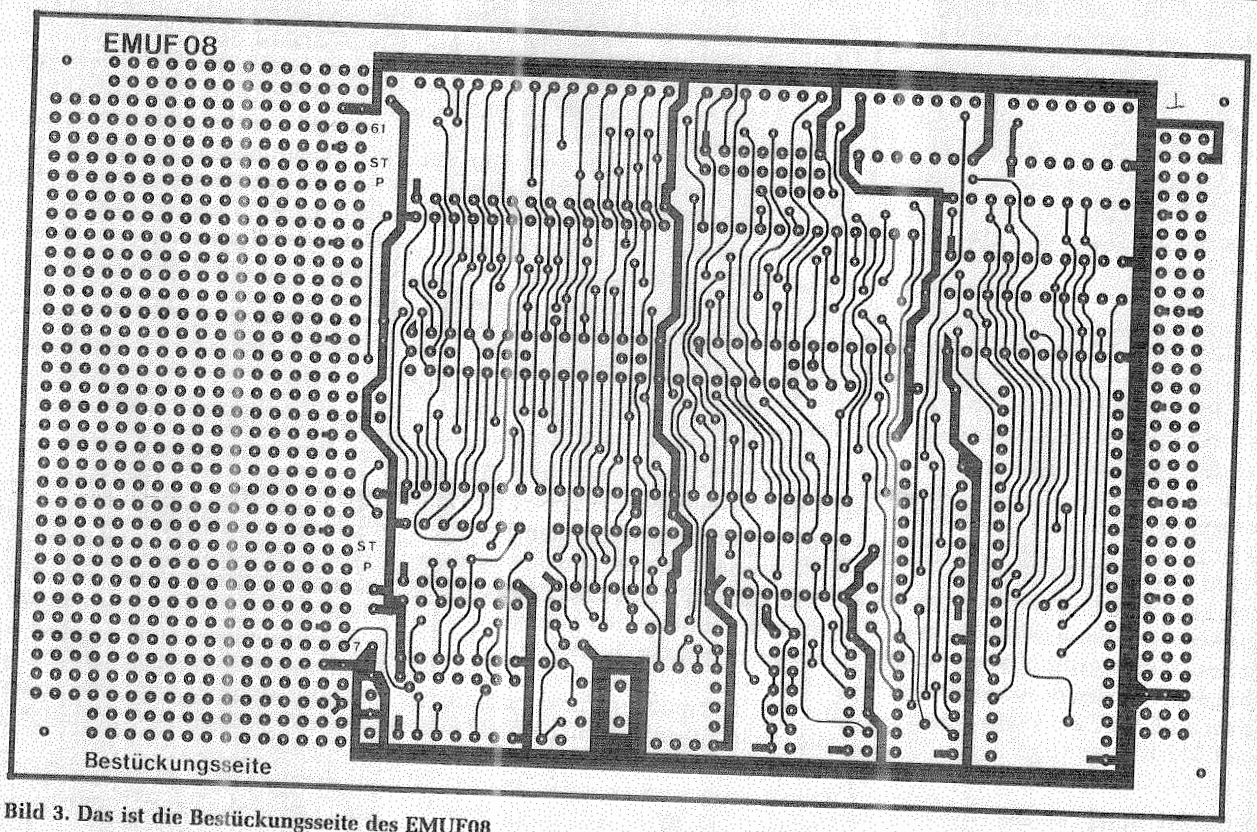


Bild 3. Das ist die Bestückungsseite des EMUF08

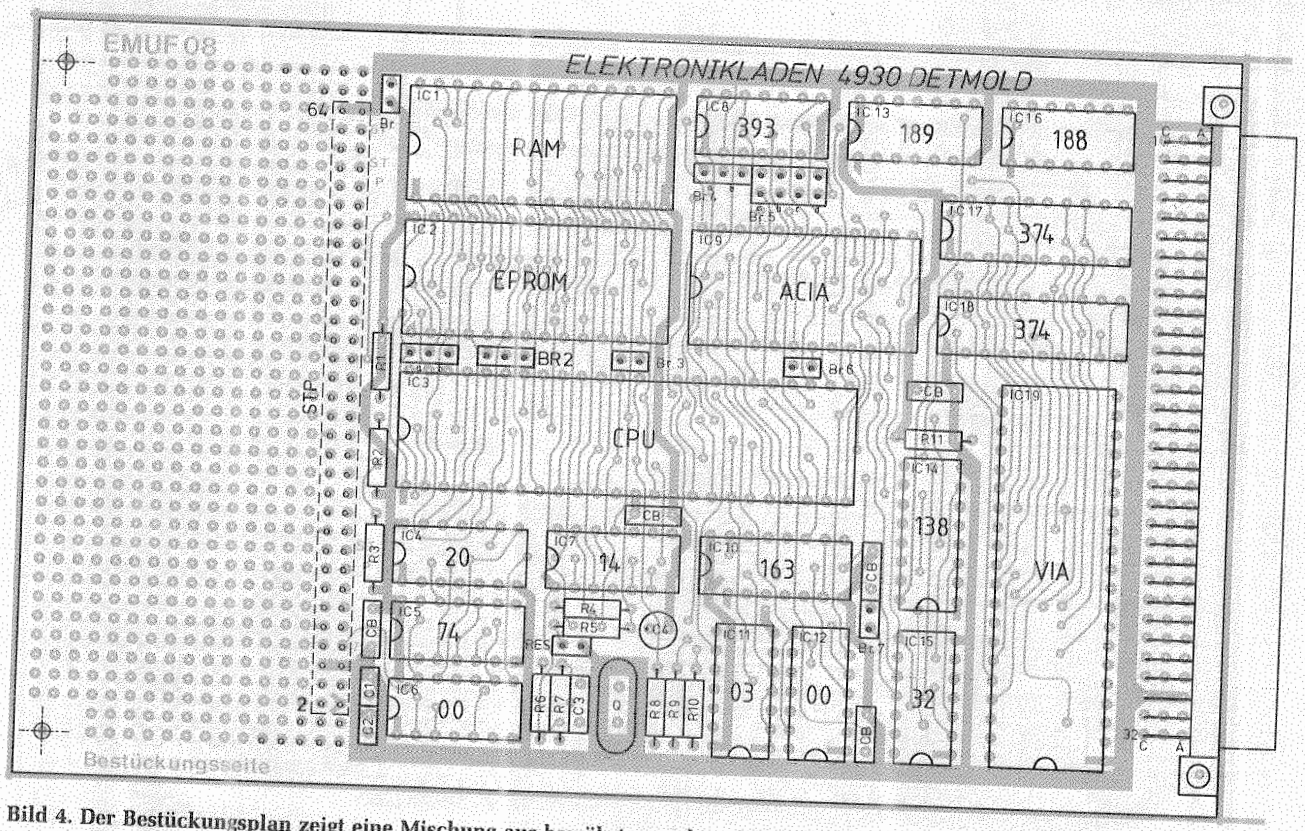


Bild 4. Der Bestückungsplan zeigt eine Mischung aus bewährten und modernsten Bauelementen

Tabelle 1: Stückliste zum EMUF

IC1	RAM	8 oder 32 KByte (max. 250 ns)
IC2	EPROM	8...64 KByte (max. 250 ns)
IC3	CPU	MC68008, 8 MHz
IC4	LS/HCT	7430
IC5	LS/HCT	7474
IC6	LS	7400
IC7	LS/HCT	7414
IC8	LS/HCT	74393
IC9	ACIA	6850, 1 MHz
IC10	LS/HCT	74163
IC11	LS/HCT	7403
IC12	LS/HCT	7400
IC13		75189 o. 1489
IC14	LS/HCT	74138
IC15	LS/HCT	7432
IC16		75188 o. 1488
IC17	S/AS	74174
IC18	S/AS	74174
IC19	VIA	6522, 1 MHz
R1	¼ Watt, 5%	1 kΩ
R2		1 kΩ
R3		1 kΩ
R4		1 kΩ
R5		22 kΩ
R6		3,3 kΩ
R7		1,5 kΩ
R8		1 kΩ
R9		1 kΩ
R10		1 kΩ
R11		1 kΩ
C8	Vielschicht	0,1 µF
C1	Schulte	22 pF
C2	Schulte	22 pF
C3	Schulte	10 nF
C4	Elko	10 µF
Q	Quarz	16 MHz

Tabelle 2: Belegung der VG-Leiste

	a	b	c		a	b	c
1	+12 V	−12 V	−12 V	17	–		
2	GND	Akku	Akku	18	–	–	–
3	RTS	TXD	TXD	19	–	–	–
4	CTS	RXD	RXD	20	PB7	PB6	PB6
5	L1-1	L1-0	L1-0	21	PB5	PB4	PB4
6	L1-3	L1-2	L1-2	22	PB3	PB2	PB2
7	L1-5	L1-4	L1-4	23	PB1	PB0	PB0
8	L1-7	L1-6	L1-6	24	–	–	–
9	L2-1	L2-0	L2-0	25	–	–	–
10	L2-3	L2-2	L2-2	26	PA7	PA6	PA6
11	L2-5	L2-4	L2-4	27	PA5	PA4	PA4
12	L2-7	L2-6	L2-6	28	PA3	PA2	PA2
13	–	–	–	29	PA1	PA0	PA0
14	–	–	–	30	GND	GND	GND
15	CB2	CA2	CA2	31	–	–	–
16	CB1	CA1	CA1	32	+5 V	+5 V	+5 V

benden Testprogramm (Tabelle 4), das in ein EPROM „gebrannt" wird hilfreich sein.
Mit Hilfe eines Oszilloskopen läßt sich durch Darstellung des Verlaufs der CPU-Signale AS• und eventuell der Datenleitungen D0–D7 die Funktion des EMUF nachweisen – oder aber die Fehlerstelle kann systematisch eingekreist werden. In ganz schwierigen Fällen kann ein periodischer Reset bei der Fehlersuche helfen (siehe mc Heft 3/86, „Das doppelte Lottchen", Seite 55). Nach dem Reset ist das Start-Flip-Flop, gebildet aus zwei NAND-Gattern (IC12) gekippt, so daß das Signal „Start" 1-Pegel hat. Über ein OR-Gatter (IC15) wird der Adreßdekodierlogik vorgegaukelt, die Adreßleitung A18 sei auf 1-Pegel. Die 68008-CPU liest nach dem Reset von der Adresse 00000 den Supervisor Stack Pointer und von der Adresse 00004 den Anfangsstand des Programmzählers. Da A18 an der Dekodierlogik aber zwangsweise „1" ist, wird dieser Zugriff auf die Adressen 40000 bzw. 40004 umgelenkt, und somit werden die Anfangswerte aus dem EPROM, das ja an Adresse 40000 liegt, gelesen. Für alle EPROMs, die für den EMUF erstellt werden gilt die Sequenz aus Tabelle 5, die auf den ersten Adressen des EPROMs steht.

Der EMUF kann zum Speicherriesen werden

An Stecker P (in der Nähe des Lochrasterfeldes) liegen die wichtigsten Signalleitungen der CPU. Erweiterungen, für die das Lochrasterfeld zu klein ist oder für die eine eigene Platine gemacht wird, können über diesen Stecker erfolgen.

Dazu lötet man am besten eine 64polige Federleiste in schmaler Ausführung mit den Anschlußreihen a und b ein. Eine Erweiterung über Stecker P könnte z. B. eine RAM Karte mit einem halben Megabyte sein, für Anwendungen bei denen ein großer Datenpuffer nötig ist.

Die Adreßdekodierung des EMUF ist sehr einfach gehalten, im Prinzip erle-

Tabelle 3: Steckbrücken

Steckbrücke	Funktion	Grundeinstellung
BR1	Verbindung von V_{cc} und V_{mm}	gesteckt
BR2	EPROM-Typ 2764	b, c gesteckt
	27128	b, c gesteckt
	27256	a, c gesteckt
	27512	a, d gesteckt
BR3	IRQ2 (VIA) auf IPL0/2 68008	gesteckt
BR4	RAM CE/A13	8 KByte a gesteckt
		32 KByte b gesteckt
BR5	Baudrate ACIA	9600 a
		4800 b
		2400 c
		1200 d
BR6	IRQ1 (ACIA) auf IPL1 68008	gesteckt
BR7	DTACK• Erzeugung on/off board	gesteckt

Tabelle 4: Testprogramm

EPROM-Adresse	OPCODE	
0000	0000 0400	Stackpointer
0004	0004 0008	Start Program Counter
0008	60FE	Branch auf sich selbst

Tabelle 5: Die Starlinformationen

EPROM-Adresse	OPCODE	
0000	0000 0400	Anfangs-SSP
0004	0004 0008	Anfangs-PC
0008	46A9 0007 0000	Start-Flipflop kippen
000E	Hier beginnt das Anwenderprogramm	

digt ein Dekodierer vom Typ 74138 (IC14) die ganze Arbeit. Der Adreßraum von 00000 bis 7FFFF wird in acht 64-KByte-Blöcke aufgeteilt *Tabelle 6*. Um eine einfache Decodierschaltung zu erreichen, gehen wir mit dem Adreßraum verschwenderisch um, andererseits ist ein Adreßraum von einem Megabyte für einen EMUF nicht gerade wenig.

Die serielle Schnittstelle

Die beiden synchronen Bausteine ACIA und VIA werden über das Flip-Flop 7474 (IC5) auf den Prozessor aufsynchronisiert. Dies ist eine Methode, die es erlaubt, so bekannte Bausteine wie 6522 und 6850 einzusetzen. Anstelle dieser „Veteranen" hätte man auch etwas Moderneres nehmen können, nur leider ist es so, daß sich viele Programmierer noch vor dem komplizierten Innenleben dieser Bausteine „fürchten". Folglich wurde der EMUF mit Beliebt-, Bewährtem ausgestattet. ACIA und VIA können Interrupts auslösen, der Interruptausgang der VIA geht auf den Interrupteingang IPL0/2 der CPU und der der ACIA auf IPL1. Dies bedeutet für den Programmierer, daß die autovektorisierten Interrupts 5 und 7 von der VIA ausgelöst werden, während der Vektor 2 von der ACIA angestoßen wird. IRQ1 und IRQ2 gehen über Steckbrücken auf IPL0/2 und IPL1.

Wer möchte, kann diese Brücken trennen und über Stecker P eine eigene Logik für vektorisierte Interrupts aufbauen.

Tabelle 6: Die Adressenbelegung des EMUF

00000 0FFFF	RAM	
10000 1FFFF	frei	(CS* vorhanden)
20000 2FFFF	frei	(CS* vorhanden)
30000 3FFFF	frei	(CS* vorhanden)
40000 4FFFF	EPROM	
50000 5000F	frei	
50010	ACIA	Daten-Register
50011		Status-Register
50012		Befehls-Register
50013		Steuer-Register
50020	VIA	Daten-Register B (ORB/IRB)
50021		Daten-Register A (ORA/IRA)
50022		Datenrichtungsregister B
50023		Datenrichtungsregister A
50024		T1 Low Order Latches/Counter (T1C-L)
50025		T1 High Order Counter (T1C-H)
50026		T1 Low Order Latches (T1L-L)
50027		T1 High Order Latches (T1L-H)
50028		T2 Low Order Latches (T2C-L)
50029		T2 High Order Counter (T2C-H)
5002A		Schieberegister (SR)
5002B		Auxiliary Control Register (ACR)
5002C		Peripheral Control Register (PCR)
5002D		Interrupt Flag Register (IFR)
5002E		Interrupt Eingabe Register (IER)
5002F		Output/Input Register B
50030 5FFFF	ACIA u. VIA gespiegelt	
60000 6FFFF	74374	
70000 7FFFF	74374, Start Flip Flop	
80000 FFFFF	frei	

Die Taktrate für die serielle Schnittstelle (ACIA) wird aus dem 16-MHz-Takt des EMUF abgeleitet. Zuerst werden die 16 MHz durch einen Teiler mit dem Teilungsverhältnis 13 geschickt, 74163 (IC10) und dann nochmal durch 2, 4, 8 und 16 geteilt; über ein Feld von Steckbrücken kann die gewünschte Baudrate eingestellt werden (BR5). Für 9600 Baud bedeutet dies, daß der ACIA 16.000.000 : (13 × 2) = 615.384 Hz als Takt angeboten werden, eine Zahl die 64 mal zu groß ist. Glücklicherweise hat die ACIA noch einen internen Teiler der dann einfach per Software auf ein Teilungsverhältnis von 64 eingestellt wird, und schon stimmt unsere Baudrate. Über die Steckbrücke BR5 (1200 Baud) und ein Teilungsverhältnis von 16 in der ACIA, kann z. B. eine Baudrate von 19 200 erreicht werden.

EMUF – auch mit Gedächtnis

Falls es gewünscht wird, daß der Inhalt des RAMs bei Ausfall der Stromversorgung erhalten werden soll, so sind dafür schon Vorbereitungen getroffen. Der Adreßdekodierbaustein (IC14) kann über ein Powerfail-Signal gesperrt wer-

Tabelle 7: Stecker P

1	VCC	2	VCC
3		4	
5	GND	6	GND
7	DTACK*	8	
9	BERR*	10	CSSYN*
11	VECTIRQ* 12	CLOCK	
13		14	
15		16	IRQAKN*
17		18	VPA*
19	AS*	20	CS1*
21		22	POWER-FAIL*
23		24	CS2*
25		26	CS3*
27	RESET*	28	
29		30	HALT*
31	A18	32	A19
33	IPL1	34	IPL0/2
35	A16	36	A17
37	A14	38	A15
39	A12	40	A13
41	A10	42	A11
43	A8	44	A9
45	A6	46	A7
47	A4	48	A5
49	A2	50	A3
51	A0	52	A1
53	D6	54	D7
55	D4	56	D5
57	D2	58	D3
59	D0	60	D1
61	R/W*	62	
63	GND	64	GND

den. Da IC14 und das RAM über eine separate Stromzuleitung verfügen, können diese Bausteine „notversorgt" werden. Der Aufbau der Power-Fail-Logik und des Akkus erfolgt am besten auf dem Lochrasterfeld.

Anwendungen des EMUF, bei welchen der Anschluß von problembezogener Hardware über VIA, ACIA und die beiden 8-Bit-Ausgangslatches möglich ist, werden am besten über einen 64poligen Stecker (*Tabelle 2*) an die VG-Leiste angeschlossen, dies ermöglicht einen sauberen Aufbau. Als erste Anwendung wird ein „elektronischer Wetterfrosch" vorgestellt werden. Diese Schaltung läßt sich einfach an den EMUF über die VG-Leiste anstecken. Sie besteht aus einer mehrstelligen Siebensegment-Anzeige, die direkt (mit Widerständen zur Strombegrenzung) an die beiden Latches 74AS374 (IC17, IC18) angeschlossen wird und einem 10-Bit-AD-Wandler mit Anschluß für Temperatur-, Druck- und Feuchtesensoren.

Ein Monitor für den EMUF: MONI-E8

Der Monitor „MONI08", ursprünglich für den mc-65816-Computer (mc 3/86) geschrieben, wurde an den EMUF angepaßt und heißt nun „MONI E8". Mit Hilfe von MONI E8 kann Software für den EMUF ausgetestet und fehlerfrei gemacht werden. Der untenstehende Ausdruck *Tabelle 8* des Help-Kommandos von MONI E8 gibt einen Überblick, welche Möglichkeiten für das Debugging (Entwanzen) von Programmen vorhanden sind.

Besitzer von Computern mit einem Assembler für 68000 (natürlich auch Hochsprachen) können mit dem EMUF sehr komfortabel arbeiten. Der EMUF wird über ein V.24-Kabel mit dem Host Computer verbunden, die Programme können auf dem Host editiert, übersetzt und anschließend als Motorola-S-Record zum EMUF übertragen werden. Mit Hilfe von MONI E8 lassen sich eventuell vorhandene Fehler aufspüren. Der Vorteil einer solchen Arbeitsweise liegt darin, daß erst dann, wenn die Software zufriedenstellend läuft, ein EPROM „gebrannt" wird, und somit die Entwicklungszeit von Software für den EMUF drastisch verkürzt wird.

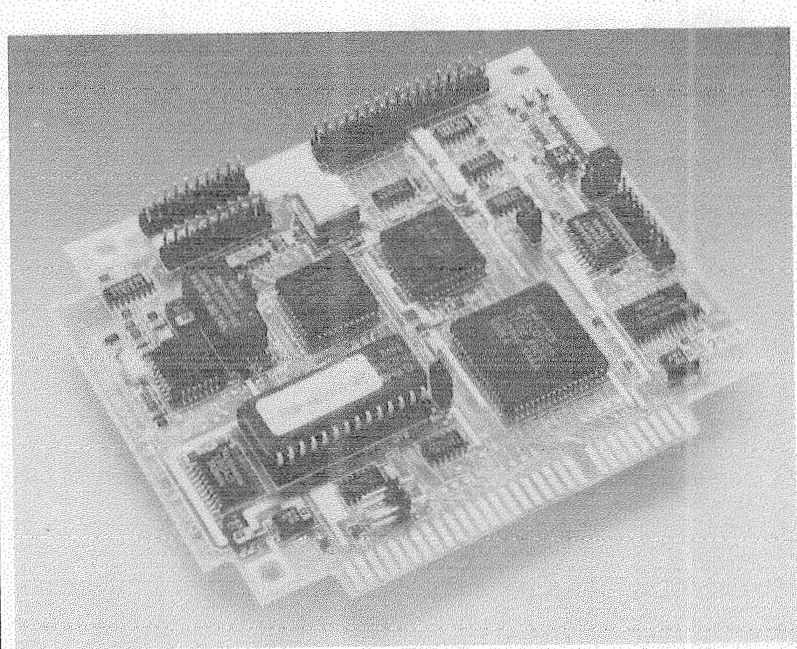
Tabelle 8: Die Monitor-Kommandos

Dump memory	: D	start (end)
Fill memory	: F	start end byte
Compare memory	: C	address1 address2 count
Move memory	: MO	address1 address2 count
Search byte	: SE	start end byte
Load s-records from 1	: L	(string)
Register display	: R	
PC display & change	: P	(value)
SR display & change	: SR	(value)
Dx display & change	: Dx	(value)
Ax display & change	: Ax	(value)
Bx display & change	: Bx	(value)
Breakpoints display	: B	
Clear all breakpoints	: CB	
Step one instruction	: S	(count)
Trace multiple step	: T	(count)
Goto	: G	(address)
Memory Test	: MT	start end
Add/Sub two hex numbers	: A	number1 number2
Calculator	: CA	string

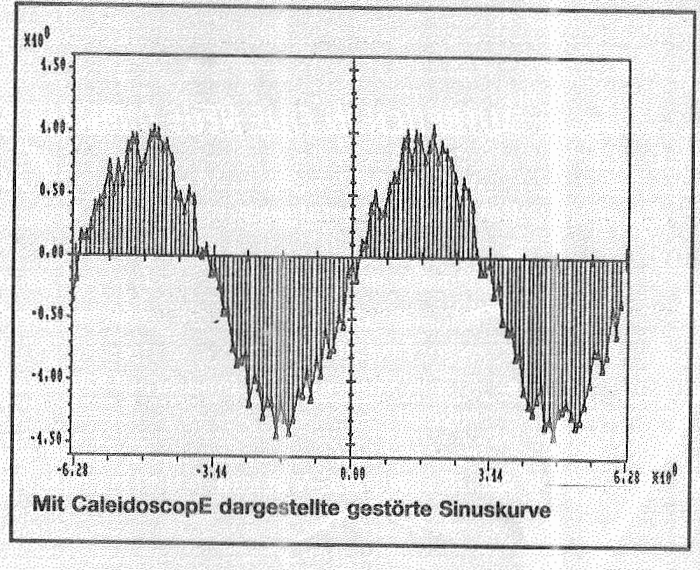
Festplatten-Controller für den PC

Zwei neue Festplatten-Controller-Karten, den WD1002-27X und den WD1002A-WX1, hat Western Digital für IBM-PCs und kompatible Computer herausgebracht. Beide Controller-Karten sind mit modernsten Bauteilen in Oberflächenmontagetechnik bestückt und benötigen daher nur einen halben Erweiterungsplatz. Die WD1002A-WX1-Karte arbeitet mit dem MFM-Codierungsverfahren (MFM = Modified Frequency Modulation), die WD1002-27X-Karte mit dem RLL-Codierungsverfahren (RLL = Run Length Limited). Mit der RLL-Codierung können im Vergleich zur MFM-Codierung bis zu 60% mehr Bit aufgezeichnet und Daten mit einer um 50% höheren Rate übertragen werden. Auf einer für RLL geeigneten Festplatte mit ST506/ST412-Schnittstelle können so bis zu 50% mehr Daten aufgezeichnet werden. Das Herz der WD1002-27X-Karte ist der aus dem weitverbreiteten Controller-Chip WD1010 entwickelte WD5011. Der WD5011 unterstützt einen 56-Bit-Fehlerkorrektur-Code und kann Daten mit bis zu 7,5 MBit/s übertragen. Durch den 56-Bit-ECC (ECC = Error Checking and Correction) werden Fehler durch Plattendefekte, die sich sonst wegen der hohen Aufzeichnungsdichte ergeben könnten. erkannt und korrigiert. Die WD1002A-WX1-Karte hat ein automatisch konfigurierbares Standard-BIOS-EPROM, so daß der Benutzer über die PC-Tastatur die Laufwerk-Parameter (Zahl der Zylinder und der Köpfe) eingeben kann. Mit diesem BIOS kann ein einzelnes Laufwerk in zwei logische Einheiten aufgeteilt werden. Der Fehlerkorrekturcode ist bei dieser Karte – für die MFM-Codierung völlig ausreichend – 32 Bit lang.

Nach Unterlagen von Western Digital



So kompakt ist die Festplatten-Controller-Karte WD1002-27X aufgebaut



Mit CaleidoscopE dargestellte gestörte Sinuskurve

Die Grafikmaschine

Ein völlig modular aufgebautes Grafiksystem unter dem viel versprechenden Namen CaleidoscopE bietet Lauer & Wallwitz für den IBM-PC und kompatible Computer an. Das Paket besteht aus den drei Bestandteilen Treiber, High-Level-Routinen und Hochsprachen-Schnittstellen. Der Treiber ist vollständig im 8086-Assembler geschrieben und vereinigt alle hardwareabhängigen Routinen. Es werden CGA-, Hercules-, EGA- und Wang-Karten unterstützt. Die High-Level-Routinen sind ebenfalls in Assembler geschrieben und bieten von einfachen Grafik-Primitiven bis hin zu Spezialfunktionen wie z. B. einer Kurvenglättung mit Savitzky-Golay-Filter oder Bezier-Kurven mit 64-Bit-Integer-Arithmetik vielfältige Grafik-Funktionen. Für Textausgaben werden Fonts in beliebiger Größe und Gestalt zur Verfügung gestellt. Die Hochsprachen-Schnittstellen unterstützen sowohl linkfähige Sprachen wie C oder MS-Pascal als auch Turbo-Pascal. Auch mit grafischen Applikationen wenig vertraute Programmierer können mit CaleidoscopE Grafik-Systeme entwerfen. Das Programm kostet 378 DM.

Lauer & Wallwitz
Erlkönigweg 9
6200 Wiesbaden
☎ 0 61 21/4 27 71

Meß- und Steuersystem als Zusatz

Jeder Rechner mit einer V.24- oder IEC-Schnittstelle eignet sich als Hostcomputer für das kleine Multifunktions-Meßdateninterface PCI 6480, das vier Analogeingänge mit programmierbaren Meßbereichen und einer Auflösung von 12 bzw. 16 Bit, ein digitales Filter, eine Grenzwertüberwachung sowie zwei Analog- und zwei Relais-Schaltausgänge für Steuer-

HW ELEKTRONIK
2000 Hamburg 19 Eimsb. Chaussee 79
Telefon 040/439 68 48 u. 430 00 19
Ihr Elektronik-Fachgeschäft in Hamburg
EIN PREISVERGLEICH LOHNT!

TEAC-LAUFWERKE
FD 35 FN 309,—
FD 35 GF/N 399,—
FD 55 A-V 349,—
FD 55 B-V 285,—
FD 55 F-V 334,—
FD 55 GF-V 385,—
SD 510 1189,—
SD 520 1269,—
SD 540 2298,—

NEC-LAUFWERKE
FD 1053 339,—
FD 1055 319,—
FD 1135C 339,—
FD 1155C 324,—
FD 1165 1189,—
FD 1035 249,—
D 3126 1289,—
D 5124 899,—
D 5126 1099,—
D 5146 1869,—

PREH-COMMANDER
PC-IL 322,50
CD 102 EX 469,—
CD 102 P 495,—
AK 87 UNI 329,—
AK 87 (APPLE) 329,—
AK 87 (APPLE) o. Block 239,—

ERSA MS 300 115,—
ERSA MS 6000 145,—

MONITORE, 22 MHz, geätzt
9" grün 269,—
9" orange 275,—
12" grün 279,—
dto. f. IBM 289,—
12" orange 285,—
dto. f. IBM 299,—

3M-Disketten „The Name" 10 St.
3,5" SS/DD 57,90
3,5" DS/DD 69,80
5,25" SS/DD 29,70
5,25" DS/DD 30,90
5,25" SS/DD 96 TPI 42,90
5,25" DS/DD 96 TPI 44,50
5,25" DS/HD 96 TPI 75,90

Exp.-Karte f. „IBM-PC" o. ä.
Epoxyd, ca. 4 k Löcher 39,50

EXP.-Karte f. „APPLE" o. ä.
2seitig, Epoxyd 14,50

QUALITÄTS-QUARZE
32,768 kHz 0,95
1,0/1,008 MHz je 8,10
1,8432/2,0 MHz je 4,40
2,048/2,097152/
2,4576 MHz je 4,90
3,0/3,2768/3,579545 je 2,70
3,6864/4,0/4,0636/4,194304/
4,433618/4,9152/5,0/
5,0688/5,12/5,185/
5,2/6,0/6,4/6,5536/
7,3728/8,0/8,872238/
9,216/10,0 MHz
10,24/12,0/14,0/14,318/
14,7456/15,0/16,0/
18,0 MHz je 1,60

QUARZ-OSZILLATOREN
4/5/6/8/10/12/14/16 MHz je 7,85

KERAM. KONDENSATOREN
RM 5 mm
100 nF/50 V % 21,—
ab 500 % 20,—
ab 1000 % 16,—

Printerkabel f. IBM o. ä.
2 m/Centr./SUB-D 20,90

ECB-Bus
10 St.-pl. 36,50
DTO, m. 10 BL 64p 65,90

IC-QUALITÄTSSOCKEL
DOPPELFEDER
8—40polig p.P. 0,02

PRÄZISIONSSOCKEL
vergoldet, gedreht
8—64polig p.P. 0,05

PRÄZISIONSSOCKEL
WIRE-WRAP/
vergoldet, gedreht
8—40polig p.P. 0,20

DIL-STECKER
f. Flb-KABEL/anpreßb.
DIS 14 1,30 DIS 28 2,20
DIS 16 1,40 DIS 40 2,60
DIS 24 2,— DIS 20 2,10

FLOPPY-STECKER verg.
10polig 3,30
20polig 3,95
26polig 4,50
34polig 4,50
40polig 5,60
50polig 7,30
60polig 11,50

CENTRONICS-STECKER verg.
14polig/löt 4,25
24polig/löt 4,50
24polig/quetsch 11,50
36polig/löt 3,25
36polig/quetsch 5,70
50polig/löt 7,30
50polig/quetsch 21,95

CENTRONICS-BUCHSEN verg.
14polig/löt 5,50
24polig/löt 4,40
24polig/quetsch 16,50
24polig/90° 9,50
36polig/löt 4,50
36polig/quetsch 5,90
36polig/90° 9,90
50polig/löt 7,40
50polig/quetsch 23,50

CENTRONICS-KUPPLUNGEN
36polig/löt 6,90
50polig/löt 7,90

VG-STECKVERBINDER verg.
64p Stecker A/C 2,50
96p Stecker 3,80
64p Buchse A/C 3,75
64p Buchse 90° 8,95
64p Buchse Wire Wrap 4,30
64p Buchse/quetsch 11,50
96p Buchse 5,95
96p Buchse Wire Wrap 6,50

DIREKTSTECKER/SLOTS verg.
2x 18p RM 2,54 3,50
2x 22p dto 3,90
2x 23p dto 4,70
2x 25p dto 3,50
2x 31p dto 4,40
2x 6p RM 3,96 2,40
2x 12p dto 2,90
2x 18p dto 5,50
2x 22p dto 5,95

SUB-D-STECKVERBINDER verg.
STECKER

Pole	löt	90°	quetsch
9	1,10	2,40	5,20
15	1,50	3,50	6,50
19	3,75	—	—
23	3,90	—	—
25	1,70	5,40	7,50
37	3,50	7,90	10,90
50	4,75	10,50	34,50

BUCHSEN

9	1,25	2,90	5,50
15	1,75	4,50	6,80
19	3,90	—	—
23	4,00	—	—
25	1,80	6,60	7,70
37	3,75	9,90	12,20
50	6,90	15,50	35,50

PFOSTEN-VERBINDER 2reihig verg.; Auswerfer/Zugentl.

STECKER-/90°		BUCHSEN
10p 1,40	1,65	1,20
14p 1,50	1,80	1,60
16p 1,65	1,95	1,70
20p 2,15	2,50	2,10
26p 2,50	2,95	2,30
34p 3,30	3,75	2,90
40p 4,00	4,30	3,50
50p 4,90	5,20	4,20
60p 5,90	6,10	5,50

FLACHBANDLITZE grau 14—64p
p. Ader/m —,10

PFOSTENFELD-VERBINDER
trennbar
36p Stecker 1reihig 1,60
36p Stecker dto. 90° 1,95
72p Stecker 2reihig 3,30
72p Stecker dto. 90° 3,75
36p Buchse 1reihig 3,90
36p Buchse dto. 90° 4,70
72p Buchse 2reihig 7,25
72p Buchse dto. 90° 9,25
JUMPER r/sw 10 St. 1,80

74 LS, 74 LS, 74 LS, 74 LS.
00/02/03 je 0,45
01/04/05/08/09/10/11/12/
27/51/54/55/133/260 je 0,55
13/15/20/21/22/26/28/30/
32/33/37/38/136 je 0,70
14/40/73/74/75/76/78/107/
109/112/113/114/122/226/
365/366/367/368/386/ je 0,95
06/07/42/86/90/91/92/93/
125/126/132/137/138/139/
279/375 je 1,25
48/49/83/85/95/96/123/
151/153/155/156/157/159/160/
161/162/163/164/173/174/175/
193/194/195/196/197/251/253/
257/258/283/290/293/352/353/
390/393 je 1,45
47/145/169/170/190/191/
192/221/240/241/242/243/244/
247/248/249/259/273/295/298/
373/374/377/378/379/395/396/
490/540/541/640/641/642/643/
44/668/669 je 1,65
165/166/245/280/348/645 je 1,85
63/147/148/399/399/670 je 2,40
154/181/183/624/625/626 je 3,50
275/321/323 je 4,95
673/674/682/683/685/688/
89 je 6,60
686/687 je 8,30
590/591/592/595/596/597 je 8,95
593/598 je 11,50

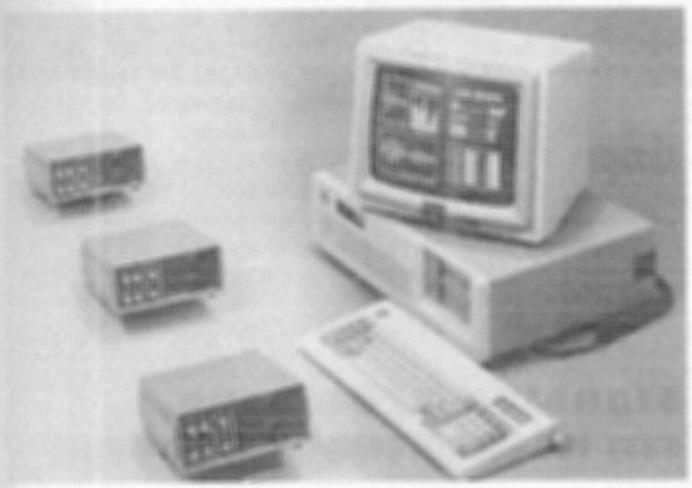
● HW-NEWS ● HW-NEWS ●

7001C	8,50	7011C	8,50
7002C	11,50	70208-8	139,—
7003C	21,90	70216-8	149,—
7004C	8,50	8052AH	99,—
6582			47,50
901225-01			28,90
901226-01			32,—
901227-03			32,—
901229-05			32,30
906114-01			23,50
8502			38,—
8701			23,—
8721			49,—
8722			49,—
8563 R9			109,—
8566 R3			84,—

★★★★★ **SERIE SMD** ★★★★★
div. aktive u. passive Bauelemente ab Lager!!!

µP µP µP µP µP	
2716-350	8,90
2716-450	7,50
2732A 250	9,25
27C32-450	15,50
2764-250	6,20
27C64-250	8,90
27128-250	7,05
27256-250	9,75
27C256-250	15,50
27512-250	28,90
2114L-200	2,95
4116-200	3,10
4164-120	3,65
4164-150	2,30
4164-150 autom.	4,70
41256-120	6,70
41256-150	6,20
41416-12	5,70
41464-12	9,95
43256-12	34,50
51C256-12	13,50
51C256-15	11,90
6116LP3	4,15
6116LFP3	7,50
6264LP15	6,50
6264LFP15	11,75
62256LP12	39,—
62256LFP12	59,—
6303	23,90
6309	37,50
6321	11,50
6340	15,40
6345	33,50
6350	11,50
63265	49,—
63450-8	188,—
63463-8	197,—
64180-4	63,—
65SC816-4	82,—
6501Q	38,50
6501AQ	42,50
6502	7,50
6502A	8,20
65C02	15,50
65SC02	14,50
65SC02-2	16,90
6504	8,50
6504A	11,90
65SC04	16,90
65SC04-2	17,90
6510	18,50
6511Q	38,50
6511AQ	42,50
6520	3,70
6520A	6,30
65SC21	15,80
65SC21-2	17,40
6522	7,50
6522A	8,75
65SC22	18,30
65SC22-2	17,70
6526	17,90
6532	7,50
6532A	11,50
65SC32	21,30
65SC32-2	23,50
6545	11,50
6551	7,50
6551A	8,20
65SC51	17,30
65SC51-2	20,70
6569	58,90
6581	36,50
6592	37,30
6800, 02 je	7,90
6809	13,90
68A09	17,50
6810	4,30
6821	3,95
6821A	4,85
6840	8,90
6845	8,50
6845B	17,90
6850	3,90
68000-8	31,50
68HC000-8	89,—
68000-10	58,—
68000-12	88,—
68008-8	32,50
68010-8	146,—
68230-8	18,50
68483	129,—
68564N04	89,—
68564N05	89,—
68590	229,—
68591	93,—
68652	28,—
68661/681 je	16,50
68020-12	598,—
68881-12	598,—
68901	53,—
7910,11 (Mod.) je	58,—
8035	5,90
8039	6,50
8080A	9,90
8085A	5,90
8086	19,90
8087-2	439,—
8088	16,90
80287-8	729,—
8155	5,20
8212	3,50
8214	6,80
8216	3,90
8224	5,15
8226	6,90
8228	8,70
8237	11,90
8243	4,50
8250	23,50
8251, 53	5,50
8255	4,95
8257, 59, 79 je	5,60
8282, 84 je	7,90
8286, 87 je	8,50
8288	13,90
8289	29,50
8748 HD	24,50
8749 HD	24,90
8755 AD	23,90
ADC 0804	10,75
ADC 0808	24,50
AM25LS2538	7,90
AM26LS31/2/3 je	4,50
AY-3-1015	10,90
AY-3-8910	17,75
AY-3-8912	16,25
EF 9364	17,25
EF 9365, 66 je	66,50
EF 9367	69,90
FDC 9229 BT	22,50
HD 4702	23,90
ICL7660/8211 je	6,75
MAX 232	17,50
MC 1488, 89 je	1,30
MC 3242P	17,50
MC 3470N	7,50
MC 3486P	3,95
MC 3487P	3,95
MSM 58321	8,25
N 82 S 23/123 je	4,25
N 82 S 126/129 je	5,40
NS405A12N	89,90
PAL 16 L8	9,90
RTC 58321	10,90
SAB 1791/3/7 je	19,90
SAB 2793/97 je	24,50
SY 2149-2	11,50
TBP 24S10/SA10 je	6,50
TBP 28L22	12,50
TC511000P10	115,—
TL 497	4,60
TMS 4416-15	5,70
TMS 4464-15	9,95
TMS 4500-19	42,50
TMS 9902	19,75
TMS 9995	69,—
U 9216	13,20
Z 80A CPU C/M	7,90
dto. CTC/PIO je	12,50
dto. SIO-O	33,90
Z 80A CPU	3,30
dto. PIO/CTC je	4,25
dto. DART/DMA je	9,90
dto. SIO-O	10,50
dto. STI	24,50
Z 80B CPU	6,50
dto. PIO/CTC je	7,70
dto. DART/SIO-O je	15,50
dto. STI	35,30
Z 80H CPU	9,75
ZN 425	13,40
ZN 426	7,80
ZN 427	24,30
ZN 428	18,30
µPD 765 AC	11,90
µPD 1990	8,50
µPD 70108-8 (V20)	18,90
µPD 70116-8 (V30)	19,90
µPD 71055C	7,80
µPD 71082C	8,50
µPD 71088C	21,50
µPD 7220AD	36,50

Versand per Nachnahme ab DM 30,— + Versandkosten; Zwischenverk. vorbeh., Erf.-Ort Hmb.



Kann jedem Computer mit V.24- bzw. IEC-Bus-Schnittstelle vorgeschaltet werden: Meßdateninterface PCI 6480

und Regelzwecke zur Verfügung stellen kann. Ein interner Zwischenspeicher nimmt dabei die Meßwerte auf, er kann aber auch dazu dienen, eine über den Computer einprogrammierte Wertetabelle als Analog-Kurvenverlauf an den beiden Ausgängen auszugeben – damit wäre ein freiprogrammierbarer Funktionsgenerator realisiert. Die beiden Relais-Schaltausgänge sind mit der Grenzwertüberwachung gekoppelt und stellen aktive Reaktionsmöglichkeiten auf festgelegte Eingangszustände dar. Einzusetzen ist das Meß- und Steuersystem unter allen gängigen Programmiersprachen. Diverse Demo-Programme für den Betrieb am IBM-PC, HP-86/9816 oder Commodore-PC erleichtern die Einarbeitungsphase. Der Komplettpreis beträgt knapp über 3000 DM, gegen Aufpreis ist ein spezielles Auswerte-Programmpaket inklusive Grafik-Routinen erhältlich.

Datalog GmbH
Bahnhofstr. 26
4010 Mönchengladbach 2
☎ 0 21 66/4 60 82

Roboterbetrieb mit PC

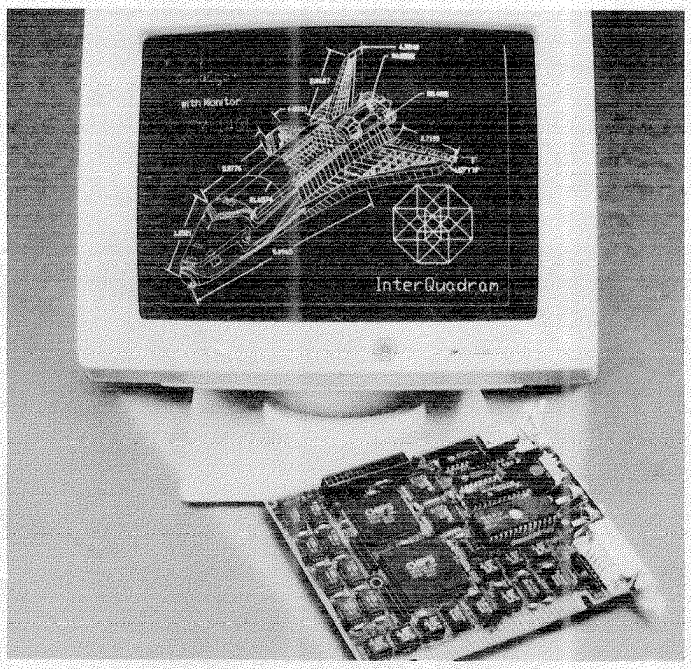
Eine komplette Robotersteuerung auf einer kurzen Zusatzplatine für den IBM-PC bietet Delec an. Das ganze System dient dazu, einen sogenannten „Teach-Robot" von der Tastatur aus zu steuern. Komplette Bewegungsabläufe lassen sich speichern und beliebig abrufen. Daß der Robot-Controller keineswegs nur Experimentiercharakter hat, zeigen einige technische Daten: Steuerung von acht Gleichstrom-Motoren in zwei Drehrichtungen, als Alternative der Betrieb von vier Schrittmotoren, integrierte Endstufen mit 5- bzw. 12-V-Ausgang, acht Digital-Steuereingänge (Lichtschranken, Endschalter usw.). Da auf der Zusatzplatine alles komplett untergebracht ist, entfällt eine separate Stromversorgung. Allerdings muß das Computer-Netzteil die Motorströme mit aufbringen.

Delec Elektronik GmbH
Dieselstr. 30
6332 Ober-Mörlen
☎ 0 60 02/14 30

Das „gute Bild" zum Nachrüsten

Die Auflösung mancher Standard-PC-Bildschirme läßt zu wünschen übrig. Das macht sich besonders störend bemerkbar, wenn man täglich mehrere Stunden davor sitzt. Mancher Anwender hätte gerne auch etwas Farbe in den grauen Monochrom-Alltag gebracht. Wer in diesem Sinne eine Aufrüstung vornehmen möchte, sollte sich für eine Paketlösung interessieren die sich HR-Set nennt und aus einer Grafikadapter-Einsteckkarte QuadEGA+ (QC 8601) und einem dazu passenden 14-Zoll-Farbmonitor vom Typ CM-1401 besteht. Das Ganze liefert eine hochauflösende Farbgrafik mit 640 x 350 Bildpunkten bzw. im Monochrom-Modus 720 x 348 Pixel. Die Bildschirmfläche wird immer vollständig genutzt, da eine Justierelektronik für exakte Bildgrenzen sorgt. Der Preis für das HR-Set: 3298 DM.

Interquadram Computer GmbH
Hermannstr. 52
6078 Neu-Isenburg
☎ *0 61 02/1 70 95*



Bringt hochauflösende Farbgrafik in den Monochrom-Alltag: HR-Set von Interquadram

Videointerface für den PC 1500/A

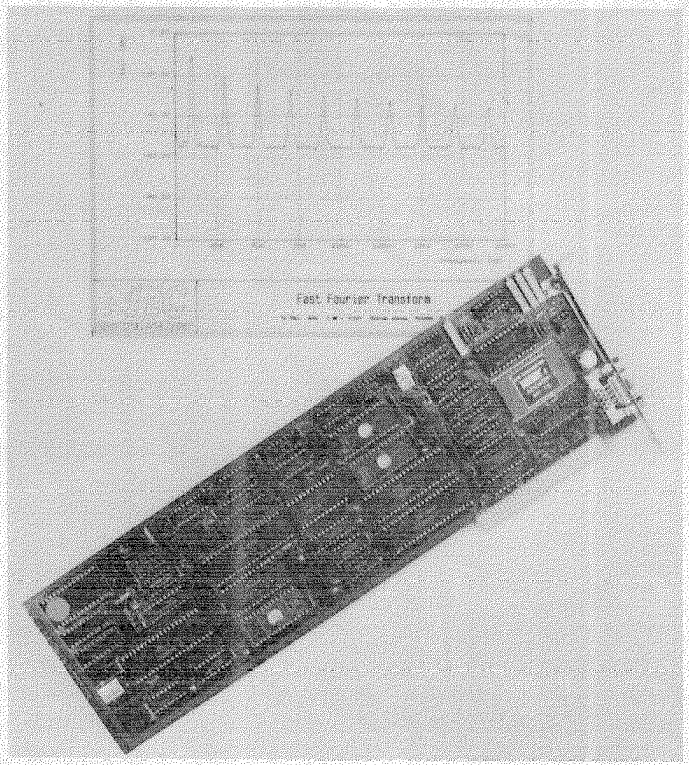
Wer sich mit den Taschencomputern PC 1500/A oder PC 1600 von Sharp beschäftigt und nicht so recht mit der kleinen LC-Anzeige zufrieden ist, kann mit einem Interface von Holtkötter auf einen Datensichtmonitor ausweichen. Auf dem Bildschirm werden bis zu 25 Zeilen mit wahlweise 40 oder 80 Zeichen je Zeile dargestellt. Dadurch wird das Ändern von Programmen mit den Cursortasten zu einer einfachen und übersichtlichen Angelegenheit. An das Interface, das 699 DM kostet und parallel zum CE 158 verwendbar ist, werden FBAS-Monitore ab 15 MHz, BAS- und auf Wunsch auch RGB-Monitore angeschlossen. Kassettenoperationen laufen mit dem Videointerface, an das sogar ein Lichtgriffel angeschlossen werden kann, bis zu 14mal schneller ab. In dem nur 16 cm × 11 cm kleinen Gehäuse (4 cm hoch) sind 2 KByte Video-RAM, 2 KByte Grafik-RAM und ein 8 KByte umfassendes Betriebssystem untergebracht. Alle gängigen Software-Produkte für die PC 1500/A und PC 1600 können uneingeschränkt weiter benutzt werden.

Holtkötter
Albert-Schweitzer-Ring 9
2000 Hamburg 70
☎ *0 40/6 69 81-0*

Signalverarbeitung, ein hochaktuelles Thema

NEC kann dem IBM-PC-Anwender mit dem Dispro-Paket zu einem kompletten Arbeitsplatz für die digitale Signalverarbeitung verhelfen. Zwei auf einer Einsteckkarte untergebrachte Signalprozessoren vom Typ µPD7720 erlauben die gleichzeitige Realisierung von digitalen Filtern sowie von Fouriertransformations-Algorithmen; anfallende Ergebnisse im Zeit- bzw. Frequenzbereich stehen für die weitere Verarbeitung bzw. die Anzeige auf dem Monitor bereit. Kontrollparameter und Daten werden von einem auf der Karte zusätzlich installierten 8085 übertragen. Im Filterteil kann man IIR- wie auch FIR-Typen, deren Koeffizienten von der Dispro-Software generiert und geladen werden, darstellen. Die Fouriertransformation schließlich erzeugt entweder eine zeitlich lückenlose 128-Punkte-FFT oder eine lückenbehaftete 512-Punkte-Auswertung.

NEC Electronics Europe GmbH
Oberrather Str. 4
4000 Düsseldorf 30
☎ *02 11/65 03 01*



Hochkarätige Fourieranalyse und digitale Filterung mit der 7720-FFT-Karte und Dispro-Software

Modular aufgebauter Industrie-I/O-Prozessor

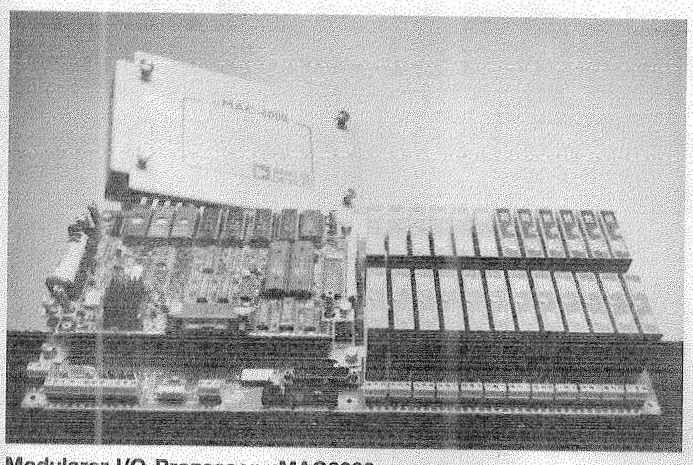
Der modular aufgebaute industrielle I/O-Prozessor µMAC6000 von Analog Devices ist ein programmierbares System, das in Basic oder C programmiert wird. In Verbindung mit µMACBasic-Programmen und bei 12-Bit-A/D-Wandlung erzielt der µMAC6000 eine Abtastrate von 1000 Erfassungen pro Sekunde, mit C-Software von 5000 Erfassungen pro Sekunde. Auf einer einzigen Platine sind 48 digital I/O, 24 analog I/O zum direkten Anschluß von Sensoren und die CPU mit Peripherie aufgebaut.

Der Prozessor eignet sich zum Steuern, Regeln oder Überwachen von Maschinen oder Anlagen. Auf lokaler Basis führt der Computer Steuerabläufe durch, prüft die Einhaltung vorgegebener Grenzwerte, tauscht Parameter mit dem Host aus und ermöglicht den Anschluß einer lokalen Bedienerschnittstelle. Im CPU-Gehäuse sind der Prozessor, der Speicher, die Kommunikations-Ports, die A/D- und D/A-Wandler und die digitalen Schnittstellenschaltungen untergebracht. Herz des Systems ist der 8-MHz-16-Bit-Prozessor 80188. In einen zusätzlichen freien Steckplatz kann der Gleitkomma-Coprozessor 8087 eingesetzt werden. Integriert sind ferner Bausteine zur 14-Bit-D/A-Wandlung und 12- oder 14-Bit-A/D-Wandlung. Hinzu kommen eine batteriegepufferte Echtzeituhr, 64-kByte-PROM als Speicher für Applikationsprogramme und ein batteriegepuffertes 256-kByte-RAM für den Standalone-Einsatz.

Zu den Besonderheiten des µMAC6000 gehört der EEPROM-Speicher mit einer Kapazität von 1 KByte, der vom Anwender zur Speicherung von Parametern verwendet werden kann. Über die RS-422-Schnittstelle wird die Verbindung zum Host hergestellt, um Programme einzulesen oder Daten zurückzuübertragen. Zwei RS-232- und eine IEEE-488-Schnittstelle dienen dem Anschluß von Druckern, Terminals und Datenerfassungseinheiten.

Das Gehäuse des µMAC6000 hat die Abmessungen von 9 × 7 Zoll. Es eignet sich für Einschubmontage. Die Computerplatinen befinden sich in dem metallenen CPU-Gehäuse, während die Signalverarbeitungsmodule Plastikgehäuse haben. Das System hat keine beweglichen Teile, ist beständig gegen Stöße und für Luftfeuchtigkeiten zwischen 5 und 95 % (nicht kondensierend) sowie für Temperaturen von 0...+60 °C vorgesehen. Zum Betrieb wird eine Spannung von +5 V benötigt.

Analog Devices
Edelsbergstr. 8–10
8000 München 21
☎ 0 89/57 00 50



Modularer I/O-Prozessor µMAC6000

Dipl.-Ing. Wolfgang Eitze
Computer Vertrieb
PC-Peripherie für den professionellen Anwender

ABECO AT-MODELL BASIS 1 **DM 3658.–**
- Gehäuse mit Platz für 5 slimline Drives
- 80286 Prozessor, 6/8 MHz Taktfrequenz
- 80287 optional
- 512 KB-RAM, aufrüstbar bis 1 MB
- 200-W-Netzteil
- 8 Steckplätze
- lizensierte BIOS
- 1,2-MB-Floppy-Laufwerk mit FD-Controller
- Monochrom-Grafikkarte
- serielle und parallele Schnittstelle
- deutsche Tastatur
- 14-Zoll-Monitor, grün oder bernstein

ABECO AT-MODELL PROFI 1 **DM 4935.–**
- wie Modell Basis 1, ohne FD-Controller
- zuzüglich 20-MB-Festplatte, 65 ms
- Harddisk/Floppy-Controller

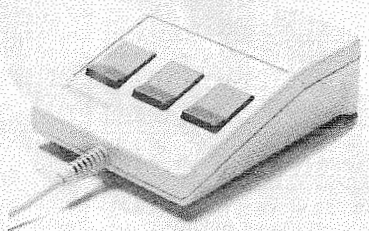
BETRIEBSSYSTEM DOS 3.2 deutsch **DM 285.–**

WEZ bietet eine komplette Palette von Erweiterungsprodukten für PC/XT/AT und Kompatible. Alle Produktnamen wie PC/XT/AT/DOS ABECO und sonstige sind geschützte Warenzeichen ihrer Hersteller.

WEZ COMPUTER VERTRIEB

Kleeweg 6
4152 KEMPEN 3
Telefon 02845-8118

LOGIMOUSE® C7
399,– DM
Händleranfragen erwünscht



- ★ Schweizer Präzisionsprodukt (siehe c't 4/86, S. 26, mc 4/86, S. 112)
- ★ an jeden Rechner mit RS-232 anschließbar
- ★ umschaltbar auf Emulation aller gängigen seriellen Mäuse
- ★ lauffähig mit allen gängigen mausorientierten PC-Programmen
- ★ Software zur Maussteuerung beliebiger tastaturorientierter PC-Programme (Lotus 1-2-3, WordStar, Framework, ...) liegt bei

Fragen Sie nach unseren günstigen Paketpreisen, besonders mit GEM und WINDOWS!

Großes Angebot an DFü Hard-/Software, günstige Preise für IBM-Kompatible sowie Spezialkarten

NEU: Grafikprogramm MiniCad für GRIP/Grips einfache Handhabung, für Maus oder Tastaturbedienung Ausgabe auf Drucker oder Plotter möglich **149,– DM**

Wir haben am 1. August das Lieferprogramm der Firma

RÖCKRATH MICROCOMPUTER übernommen.

Telefon (02 41) 3 28 96
Noppiusstr. 19, 5100 Aachen

KRISCHER COMPUTERTECHNIK

DER EMUF 08

mc 2/87 — mit 68008

08/BS1	Platine (Epoxy, mit Bestückungsaufdruck und Lötstoplack), allen aktiven und passiven Bauteilen, ohne RAMs und Eproms	159,00
08/BS2	wie 08/BS1, jedoch ohne 6522	146,00
08FB	EMUF08 komplett, aufgebaut und getestet	249,00
08LP	Leerplatine des EMUF08	48,00
MON 08	Software-Monitor für den EMUF08	95,00
	Handbuch einzeln: 5 DM in Briefmarken	

DER BASIC-EMUF

mc 12/86 — mit 8052

BE/BS1	Platine, GAL (8K o. 32K), Manual	98,00
BE/BS2	Platine, GAL (8K o. 32K), CPU Quarz, Manual	198,00
BE/BS3	Uhr (58321) und Akku	29,00
BE/BS4	µPD 7002, ZN 485, Quarz	29,00
BE/BS5	SN 75468, 2 Stück Relais	29,00
BE/BS6	LCD/KEY – 2 x 16 Stellen Anzeige, 16 Taster	198,00
BE/BS708	Platine, GAL8, CPU, Quarz, 8255, TL7705, MAX232, LS08, LS373, RAM8	238,00
BE/BS732	wie BS708, jdoch statt GAL8/RAM8 hier GAL32/RAM32	278,00
BE/BS808	wie BS708 + alle passiven Bauteile und Fassungen	260,00
BE/BS832	wie BS 732 + alle passiven Bauteile und Fassungen	300,00
BE/FBCPU8K	BASIC-EMUF, CPU Teil aufgeb. + getestet mit 8KRAM	398,00
BE/FBCPU32K	wie FBCPU8K, jdoch mit 32 KRAM	438,00
BE/KOM8K	BASIC-EMUF komplett, aufgeb. u. getestet mit 8KRAM	598,00
BE/KOM32K	wie KOM8K, jedoch mit 32 KRAM	638,00
BE/FBLCD	wie BS6, jedoch aufgebaut und getestet	248,00
BE/Handb.	Manual (ca. 90 Seiten) zum BASIC EMUF einzeln	15,00

RAMS für EMUF 08 + BASIC-EMUF

RAM8	8K stat, 120 us	7,90
RAM32	32K stat, 100 us	49,00

Der NDR-Klein Computer

		DM			DM
SBC 2	BS	59,00	IEC	BS	169,00
CPU Z80	BS	69,00	REL	BS	218,00
CPU 64180	BS	269,00	PROMER	BS	99,00
CPU 68K	BS	139,00	ACRTC BAS.	FB	1300,00
CPU 68000/8	BS	198,00	ACRTC COL.	FB	1300,00
CPU 68000/12	BS	398,00	BUS II (6 Pl.)	BS	87,00
CPU 68020	FB	1798,00	BUS III (12 Pl.)	BS	159,00
FDC	BS	258,00	BUS IV (18 Pl.)	BS	249,00
SER	BS	99,00	ROA 64	BS	49,00
IOE	BS	49,00	ROA PROM 1	BS	45,00
KEY	BS	49,95	DRAM 64	BS	190,00
CAS	BS	59,00	DRAM 128	BS	229,00
GDP 64K	BS	299,00	BANKBOOT	BS	69,00
HCOPY	BS	198,00	AD 8x16	BS	148,00
COL256	BS	448,00	AD 10x1	BS	265,00
SOUND	BS	89,00	DA	BS	139,50
SPRACHE	BS	398,00	HEXIO	BS	ab 129,00

NEU
LOGIKANALYSATOR LOG16
16 Kanäle/10 MHz/bildschirmorientierte Software für NDR-Klein
BS **498,00** FB **598,00** LP **298,00**
Handbuch einzeln **20,00 DM**

NEU
LEERKARTEN JETZT AUCH LIEFERBAR VON

CPU 68020 LP	398,00
ACRTC BASIS LP	358,00
ACRTC COLOR LP	358,00

WIR SIND IN DORTMUND AUF DER COMPUTER-SCHAU IN HALLE 6.

Alle angebotenen Baugruppen erhalten Sie beim

ELEKTRONIKLADEN Mikrocomputer GmbH & Co. KG
Eggestr. 70, 4930 Detmold 18, Telefon 05232/8171
8000 München 19, Schulstr. 19, Tel. 089/169977
4400 Münster, Hammer Str. 157, Tel. 0251/795125

mc-markt

CPC-Zusätze

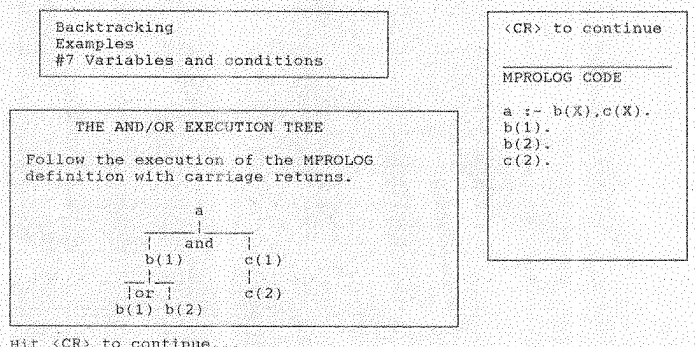
Auf einer Zusatzkarte mit EPROM-Programmiergerät, V.24-Schnittstelle und einem Erweiterungs-EPROM bietet Philosoft ein interessantes Programmpaket für die CPC-Computer an. Es umfaßt u. a. eine WordStar-kompatible, relativ schnelle Textverarbeitung einschließlich Fettdruck, Unterstreichen und Kursivschrift. Das Beachtliche an dem Programm ist, daß Hoch- und Tiefstellungen im Text auf dem Bildschirm angezeigt werden. Bei Kassettenbetrieb muß das Programm zwangsläufig auf die ausführlichen Hilfsmenüs verzichten, auch bleibt die Textlänge auf die Speichergröße des Rechners beschränkt. Außerdem bietet die Zusatzkarte einen Z80-, 8080- und 8048-Assembler, einen symbolischen Debugger, ein zu Modem 7 kompatibles Übertragungsprogramm und auch die Emulation von CP/M-Systemfunktionen. Damit ist die Entwicklung von Assemblerprogrammen bezüglich des Komforts vergleichbar mit dem Arbeiten in Turbo-Pascal bei gleichermaßen optimalen Debugging-Fähigkeiten. Die Karte ist modular ausbaubar und ab 280 DM erhältlich.

Philosoft EDV
Pariser Platz 2
8000 München 80
☎ 0 89/4 48 26 01

Prolog-Kurs: Keine Vorkenntnisse notwendig

Für Anwender eines IBM-PC – oder eines Kompatiblen – bietet Epsilon zum Preis von rund 290 DM einen Prolog-Kursus zum Selbststudium an, der ein 500 Seiten starkes Handbuch und entsprechende Teachware umfaßt. Der „Prolog Primer" ist zwar in englischer Sprache geschrieben, setzt dafür aber keine spezifischen Vorkenntnisse voraus und beginnt mit der Erläuterung wesentlicher Grundbegriffe. Durch regelmäßige Querverweise auf die begleitende Teachware soll dabei sichergestellt werden, daß der Lernende die zu jedem Themenschwerpunkt angebotenen Beispiele durcharbeitet – besonders ausgearbeitete Übungen ermöglichen die selbständige Kontrolle der Lernfortschritte. Besonderes Gewicht wird auf Fragestellungen gelegt, die über die Grundkonzepte Prologs hinausgehen; ein besonderes Kapitel ist der für die Entwicklung von lernfähigen Expertensystemen unerläßlichen dynamischen Programmierung gewidmet.

Epsilon GmbH
Kurfürstendamm 188/199
1000 Berlin 15
☎ 0 30/8 82 69 91



Mit vielen anschaulichen Übersichtsdiagrammen – alles komplett in Englisch – arbeitet der Prolog-Kurs zum Selbststudium

Nachschlagewerk und Katalog zugleich

Die bekannten RIM-Electronic-Jahrbücher stellen für den an der Hardware interessierten Computerbesitzer, aber auch für den HiFi-Enthusiasten oder Hobbyelektroniker eine nicht zu unterschätzende Arbeitshilfe dar. Sie umfassen das komplette Angebotsspektrum des Unternehmens an elektronischen Bauelementen, Bausätzen, Meß- und Prüfgeräten, Computern und -peripherie sowie Werkzeugen und bieten Informationen über ein reichhaltiges Bausatz-Programm mit über 360 Angeboten. Rund 2 kg bringt die Ausgabe für das Jahr 1987 auf die Waage; sie umfaßt 1288 Seiten, zeigt rund 3000 Abbildungen und über 600 Schaltungen, Applikationen sowie Diagramme und kostet 16 DM.

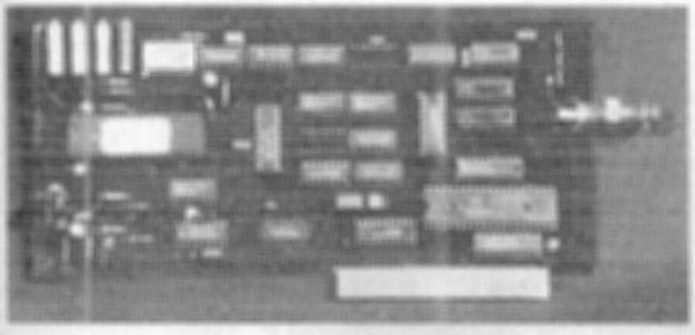
Knapp 1300 Seiten an kompilierter Information für den Elektronik- und Computer-Praktiker: RIM-Electronic-Jahrbuch 87

RIM Elektronik GmbH
Bayerstr. 25
8000 München 2
☎ 0 89/55 81 31

Video-Digitalisierer mit 256 Graustufen

Die Video-Digitalisierer-Karte VD 8003 von Kolter ist als Erweiterungskarte für den IBM-XT/AT und kompatible Computer konzipiert. Die maximale Auflösung ist 1024 x 512 Pixel mit bis zu 256 Graustufen. Ein 8-Bit-Flash-A/D-Baustein sorgt für die schnelle Digitalisierung der Bilder. Der eingebaute Zeilenspeicher hat eine Speichertiefe von 1024 x 8 Bit. Das Einlesen dauert auf einem IBM XT 980 ms, auf einem IBM AT (8 MHz) 380 ms. Helligkeit und Kontrast können eingestellt werden. Die Karte eignet sich zum Erstellen von Fotomontagen bis hin zum Aufbau von Bildbibliotheken auf Disketten. Als Kamera kann eine einfache Video-Kamera mit BAS-Ausgang verwendet werden. Die Karte kostet komplett mit Software rund 990 DM.

Kolter Electronic
Nikolaus-Ehlen-Str. 11a
5042 Erftstadt
☎ 0 22 35/7 67 07



Der Video-Digitalisierer paßt auf eine PC-Erweiterungskarte

Farsight, die BESONDERE SOFTWARE.

Wir erhielten die folgende Anfrage zu Farsight, der BESONDEREN Software:

Farsight kostet nur DM 399.90 +MWSt./Sfr. 399.90

Nun gibt es auch eine passende leistungsfähige Adressenverwaltung für DM 175.– +MWSt./Sfr. 165.–

Bezugsquellen:

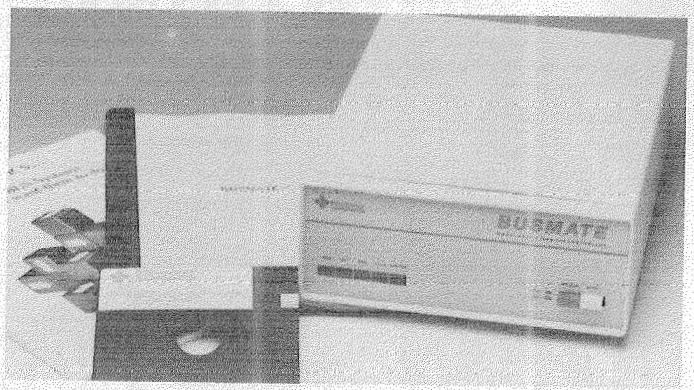
Bundesrepublik Deutschland:

Schweiz:

Österreich:

Generalvertrieb für Europa:

A. + L. Meienhofer
im Spalten 23
CH-8906 Bonstetten/ZH
Tel. (41) (1) 700 30 37



BusMate mit Arbeits- und Demosoftware erlaubt einen unkomplizierten Einstieg in die IEC-Bus-Technik

Den Computer zum Controller befördern

Das US-Unternehmen ICS hat mit BusMate ein Zusatzgerät entwickelt, mit dem sich ein Alltags-PC zum voll funktionsfähigen Controller für den IEC-Bus aufwerten läßt. Die Steuerung von bis zu 14 Geräten (Listener bzw. Talker) ist dabei möglich. Als übergeordneter Rechner kann sowohl einer aus der Kategorie der geschlossenen Systeme wie auch ein Vertreter der „offenen Architektur" fungieren, z. B. ein IBM-PC. Wie auch immer – der Rechner muß zur Verbindung mit BusMate lediglich eine serielle Schnittstelle besitzen. An der Bedienung und Transparenz des Computers ändert sich rein äußerlich durch eine solcherart realisierte IEC-Bus-Konfiguration nichts, vielmehr lassen sich sämtliche Bus-Befehle in herkömmliche Basic-, Pascal- oder Fortran-Programme einbinden. Das Gerät (etwa 2000 DM) akzeptiert die entsprechenden Anweisungen und erzeugt daraus die Steuerbefehle für die an den Bus angeschlossenen Subsysteme. Meß- oder Prüfergebnisse werden von diesen zurückgemeldet und über BusMate in den Computer eingespielt.

Meilhaus Electronic GmbH
Fischerstr. 2
8039 Puchheim
☎ 0 89/80 70 81

Sprachsynthese macht Fortschritte

Das Verfahren der Wellenform-Codierung (ADPCM) bringt gegenüber den parametrischen Methoden (Formantensynthese, LPC, Parcor) einerseits eine durchaus gewünscht hohe Wiedergabequalität, benötigt andererseits aber eine größere Datendurchsatz-Rate, und zwar 10...30 kBit/s, was aber mittlerweile durch entsprechend schnelle, hochintegrierte Speicherbauelemente in den Griff zu bekommen ist. NEC unterstützt ADPCM-Entwicklungen mit einer Familie von Sprachsynthese-ICs in stromsparender CMOS-Technologie. Lieferbar sind zur Zeit der

Begraben Sie Ihren
SCHNITTSTELLENTESTER...
...jetzt gibt es »COMIX«

»COMIX«: Die reine Softwarelösung zur Darstellung, Analyse und Steuerung von RS232/V.24-Schnittstellen nur über den Monitor!

»COMIX«: Ein deutsches Produkt der ComFood Software GmbH mit 50seitigem Handbuch!

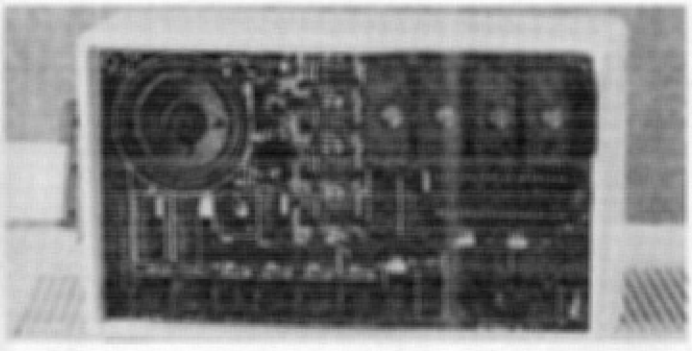
»COMIX«: Support direkt vom Hersteller!

»COMIX«: Endkundenpreis: 299,— DM

Fordern Sie gleich ein Datenblatt zu dieser einmaligen Lösung an!
Händleranfragen willkommen.

ComFood Software GmbH, Ossenkampstiege 70 A, 4400 Münster, Telefon 0251-71 97 68

– Wir füttern Ihren Computer –
ComFood Software GmbH



Die Demobox 7759 erlaubt in Zusammenarbeit mit einem PC eigene Sprachsynthese-Entwicklungen

µPD7756 mit 256 KBit ROM und der ohne eigenen Speicher auskommende µPD7759 mit 1 MBit Adreßraum. Spezialkenntnisse setzt die Sprachcodierung durch den Anwender nicht voraus; zur Entwicklung genügt bereits ein IBM-PC mit entsprechender Zusatzplatine. Für die Auswertung der Syntheseergebnisse und die In-Circuit-Emulation dient eine kleine Demobox mit der Typenbezeichnung EBIBM-7759. Sie wird komplett mit PC-Diskette und Bedienungsanleitung geliefert und ist mit den notwendigen Tiefpaßfiltern, einem Verstärker und Lautsprecher ausgerüstet.

NEC GmbH
Postfach 33 03 28
4000 Düsseldorf 30
☎ 02 11/65 03 01

Verbesserte Programme für den Atari ST

Die Programme C-Auftrag/C-Text, die jetzt seit einem Jahr auf dem Markt sind und Kunden-, Lieferanten-, Artikel/Lager- und Auftragsverwaltung enthalten, werden in einer stark verbesserten Version angeboten. Besonders die Umsetzung von Werten in Grafiken und der Einsatz von beliebig langen Textbausteinen in der Fakturierung wurden verbessert. Mit der Textverarbeitungskomponente C-Text Plus können Druckformulare und Serienbriefe erstellt werden. Für das integrierte Paket sind 570 DM zu zahlen; ein Update für die älteren Versionen kostet 35 DM. Wer mit den in C-Auftrag gespeicherten Daten gleich in die Finanzbuchhaltung einsteigen möchte, kann jetzt vom gleichen Hersteller C-Fibu bekommen.

Das Programm bietet Buchungen auf mehreren Konten, Führung von offenen Posten mit Mahnwesen und Zahlungsausgleich. Es kann auch ohne C-Auftrag als eigenständiges Programm eingesetzt werden. Wer an seinem Atari ST schon eine Festplatte angeschlossen hat, braucht auf den Backup-Befehl, der Bestandteil von MS-DOS ist, nicht länger verzichten. Für 98 DM gibt es ein Programm, das alle Dateien einer Festplatte auf Disketten sichert. Für etwa 400 DM bietet C-Soft ein EPROM-Programmiergerät an, das EPROM-Typen bis zum 27512 brennen kann.

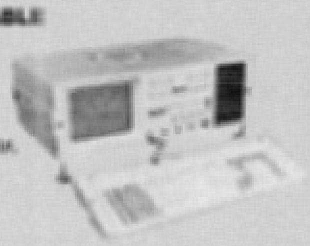
C-soft GmbH
Holzfällerstr. 4
8400 Regensburg
☎ 09 41/8 39 86

ECD-ADVANCED (AT)

MS-DOS 3.1 ... DM 50.– Aufpreis
(700 Seiten-Handbuch in Deutsch)

ECD-PROFESSIONAL (XT)

ECD-PORTABLE



LAUFWERKE/FESTPLATTEN

MONITORE (große Auswahl) z. B.

Erweiterungen XT AT

Wir sind STAR-Vertragshändler

SOFTWARE z. B.

Große Softwareauswahl verfügbar!

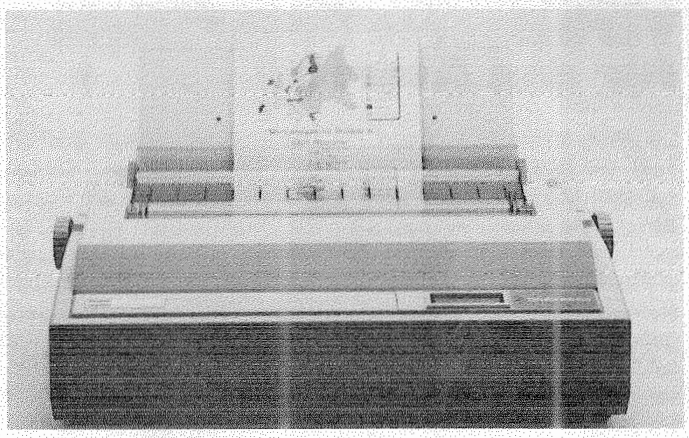
Baby-AT ... **ab 2450.–**

AT-Portable ... **ab 3690.–**

HARD- UND SOFTWARE ● SERVICE ● SCHULUNGEN

IHR PARTNER IN SÜDDEUTSCHLAND

ECD COMPUTERTECHNIK GmbH
Verwaltung/Verkauf:
Sonnenstr. 2, 8050 Freising,
Telefon 0 81 61/6 69 96



Bedienungskomfort sehr gut, Geräuschpegel in Grenzen: 24-Nadel-Hochgeschwindigkeitsdrucker von Fujitsu

Stiller Zeitgenosse

Zwar schaffen die 24 Nadeln der neuen Fujitsu-Hochgeschwindigkeitsdrucker DL 2600 und DL 2600 Color pro Minute 125 Zeilen, sie sind aber dennoch erstaunlich leise (Geräuschpegel von 55 dBA). Daß Grafik und diverse Schriftarten wählbar sind, gilt in dieser Druckerklasse als selbstverständlich – keineswegs alltäglich ist hingegen die für mehr Übersicht bezüglich der Druckparameter sorgende Flüssigkristall-Anzeige. Eine weitere Erleichterung für den Anwender ist die ebenfalls sichtbare Emulationsangabe (IBM-Graphics-Printer, JX 80, Diablo 630 usw.).

Die Color-Version liefert hochqualitativen Mehrfarbendruck und eignet sich recht gut zum Herstellen farbiger Overheadfolien. Beide Drucker sind sowohl mit paralleler als auch serieller Schnittstelle erhältlich.

Fujitsu Deutschland GmbH
Rosenheimer Str. 145
8000 München 80
☎ 0 89/41 30 10

Der diskettenlose CP/M-Computer

Ein EPROM-Boot-Paket für den CP/M-3.0-Einplatinencomputer PROF-180X von Conitec erlaubt zusammen mit einer RAM-Floppy von 400 KByte einen Betrieb völlig ohne Diskettenlaufwerk. Beim Einschalten des Rechners (der natürlich nach wie vor über Laufwerk-Schnittstellen verfügt) werden automatisch die Betriebssystem-Dateien in die RAM-Floppy kopiert. Zusätzlich sind im EPROM noch etwa 30 KByte für eigene Anwender-Files vorhanden. Das hier genannte Paket kostet lediglich 86 DM, der Einplatinencomputer selbst schlägt mit 1250 DM in der 6,1-MHz- und mit 1622 DM in der 9,2-MHz-Ausführung zu Buche.

Conitec GmbH
Grafenstr. 31
6100 Darmstadt
☎ 0 61 51/2 60 13

Schaltungsideen für Elektroniker

Arbeitserleichterungen, die es in sich haben, weil sie Entwicklungszeit und damit Kosten sparen: Die interessantesten Applikationsbeispiele aus den ELEKTRONIK-Rubriken „Schaltungspraxis", „µC-Praxis" und „Applikation" finden Sie jetzt in einer neuen Sonderheftserie. Thematisch geordnet sind sie eine einzigartige Arbeitshilfe für Ingenieure und Elektroniker. **Die einzelnen Hefte sind unter folgenden Titeln erschienen:** ● Mikroprozessorschaltungen ● Steuer- und Regelschaltungen ● Meß-, Prüf- und Sensorschaltungen ● Interface- und Wandler-Schaltungen ● Digital- und Logikschaltungen ● Oszillatoren, Generatoren, Rechen- und Filterschaltungen ● Analog-, Operationsverstärker, Stromversorgung.

Nr. 6: Oszillatoren, Generatoren, Rechen- und Filterschaltungen

● Erprobte Applikationen für RC-, Sinus- und Quarzoszillatoren, für Impuls-, Funktions-, Leistungs- und HF-Generatoren. Außerdem Beispiele für gängige Filtertypen, weiterhin Rechenschaltungen wie Differenzierer, Multiplizierer, Dividierer, Logarithmierer und Fourier-Transformation.

NEU
Elektronik-Sonderheftserie „Schaltungspraxis"

Hier gibt's die neuen Arbeitshilfen für Elektroniker:

Bei allen Bahnhofsbuchhandlungen, beim Elektronik-Fachhandel, in Mikrocomputershops, bei größeren Zeitschriftenverkaufsstellen, in Buchhandlungen oder direkt beim Franzis-Verlag

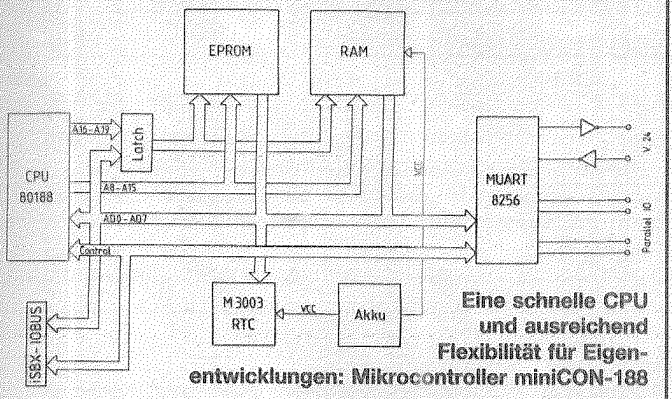
● Per Postkarte mit Hinweis „Sonderheft Oszillatoren, Generatoren, Rechen- und Filterschaltungen" (Rechnung wird zugesandt) oder

● Per Scheck (DM 22.50)

Franzis'
Franzis-Verlag, Karlstraße 37,
8000 München 2,
Tel. 0 89/51 17-2 39/-3 80

In der Schweiz: Thali AG, Fachliteratur und Software, Industriestraße 6,
CH-6285 Hitzkirch

In Österreich: Erb-Verlag,
Ges.m.b.H. & Co. KG, Mariahilfer Straße 71,
A-1061 Wien



Eine schnelle CPU und ausreichend Flexibilität für Eigenentwicklungen: Mikrocontroller miniCON-188

Anspruchsvolle Aufgaben gerade recht

Ein Prozessor 80188 mit 5,2 MHz, maximal 64 KByte RAM, bis zu 128 KByte EPROM, ein Multifunktionsbaustein 8256 und ein Uhrenbaustein tun auf der miniCON-188-Platine ihren Dienst. Alle Komponenten zusammen ergeben einen komfortablen Mikrocontroller, der auf seiner Platine ein großes Verdrahtungsfeld für eigene Applikationen bietet. Über das iSBX-Interface hat der Anwender Zugang zu zahlreichen Peripherie-Modulen, wie z.B. Schnittstellen, EPROM-Programmiergerät und A/D- bzw. D/A-Umsetzer. Auf einem PC entwickelte Programme lassen sich über die V.24-Schnittstelle in den Controller transferieren und dort abarbeiten. Phytec offeriert in diesem Zusammenhang eine interessante Dienstleistung: Kundenentwicklungen mit diversen Zusatz-Applikationen auf der Platine können in Musterstückzahlen bereits fertig ab Werk geliefert werden.

Phytec Meßtechnik
Donnersbergstr. 32
6500 Mainz 42
☎ 0 61 31/50 88 71

V.24-Prüfstand für die Jackentasche

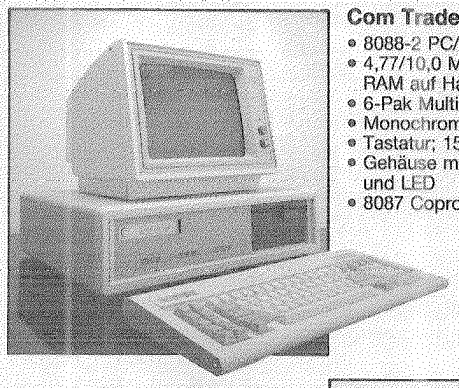
Eine eigene Stromversorgung benötigt er nicht und handlich ist er auch, der kompakte Schnittstellentester für RS-232/V.24 von Jet Computer. Der Tester wird zwischen Gerätebuchse und Anschlußkabel geschaltet und zeigt mit 50 LEDs den Pegelzustand auf den jeweiligen Leitungen an. Die Differenzierung erfolgt optisch eindeutig zwischen High-Pegel (über +3 V) und Low-Signal (unter -3 V). Auch DCE-DTE können getestet werden. Recht hilfreich bei spezielleren Prüfungen sind 50 Jumper-Buchsen, die jede einzelne Leitung zum Auftrennen, Überbrücken oder für Kreuzverbindungen zugänglich machen.

Jet Computer
Box 138
S-182 12 Danderyd,
Schweden
☎ 0 04 68/7 53 31 45

Choose it for the Best!!!

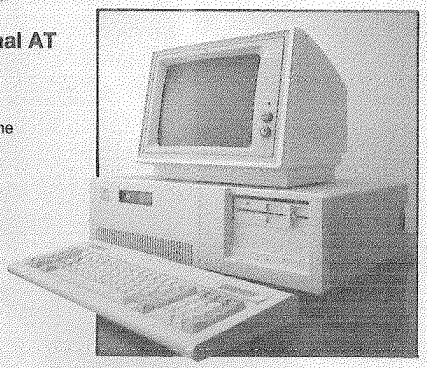
DEUTSCHE MESSE UND AUSSTELLUNGS-AG HANNOVER

HALLE 6 Stand H54



Com Trade Turbo XT
- 8088-2 PC/XT-kompatibel
- 4,77/10,0 MHz, bis zu 640 KByte RAM auf Hauptplatine erweiterbar
- 6-Pak Multi I/O-Karte
- Monochrome Grafikkarte
- Tastatur; 150 W
- Gehäuse mit Schlüsselschalter und LED
- 8087 Coprozessor als Option

Com Trade Professional AT
- 80286 PC/AT-kompatibel
- 6/10 MHz, 6/12 MHz, 0 oder 1 Wortezyklus
- 1 MByte RAM auf Hauptplatine
- Monochrome Grafikkarte (Jumper switch)
- FDD/HDD Controller
- Seriell-/Parallelkarte
- XAT-Tastatur; 200 W
- AT-Hauptplatine in XT-Format auch verfügbar



WIR SIND TAIWANS FÜHRENDER GEHÄUSEHERSTELLER

Wir machen alles: Vom Stanzen und Pressen bis zum Lackieren kontrolliert Ling Yih jeden einzelnen Schritt der Herstellung. Wir zeigen Ihnen gern unsere Fabrik – Ling Yih hat nichts zu verbergen, aber viel zu zeigen. Wir bieten nicht nur höchste Qualität, sondern auch einen annehmbaren Preis. Schreiben Sie uns noch heute. Wir liefern prompt.

ComTrade Technology Co., Ltd.
9F-3, 57 FU-HSING N. RD., TAIPEI, TAIWAN, R.O.C.

LING YIH CO., LTD.
TEL: (02)771-7166 FAX: (02)711-3506 TLX: 20232 LINGYIH

DELA MAUS

DRUCKER
STAR NL 10 . . 699.00
Der meistgekaufte Drucker!
- Matrixdrucker
- NLQ-Mode
- 120 Zeichen pro Sek. im Normalbetrieb
- voll EPSON FX kompatibel
- Bidirektionaler Druck
- Traktor und Gummiwalze
- Einzelblatteinzug
- Farbbandkassette
- Einsteckbares Interfacemodul
- seriel. Commodoreinterface für C64/C128
- Centronicsinterface
- IBM-Interface
- inkl. englisches Handbuch
- deutsches Handbuch 20.–

Bei Bestellung bitte Interface angeben!

CITIZIEN 120 D 485.00
STAR-STX80 ..... 199.00
NEC P6 1298
2-Farbdrucker f. C 64 . . 89.50

DISKETTEN
Nur geprüfte Ware aus der Produktion namhafter Hersteller!!
5"25-Disketten 10 Stück
No Name 1D 10.90
No Name 2D 13.90
3"5-Disketten 10 Stück
No Name 1DD 27.90
3"-Disketten 10 Stück
Maxwell CF2 ..... 79.90

Elektr. Bauteile
EPROMS:
2764 250ns 7.50
27128 250ns 7.50
27256 250ns 12.90

allg. Elektronik
RAMS:
4264 64K*8 Statikram 8.90
41256 256K*1 Dynamicram 8.90
4464 64K*4 Dynamicram 9.90
MICROCOMPUTER:
6502 CPU 9.80
6520 2*8Bit Port 5.90
6522-A VIA 2MHZ 11.90
6532 VIA 128Byte RAM 9.80
TTL-ICs-Superpreise
74LS00 ..... 0.60
74LS08 ..... 0.60
74LS373 ..... 1.60
Kompletfangebot: Info anfordern!
WIDERSTÄNDE 1/4 Watt
100 Stück-Pack p. Wert ..... nur 2.00
IC-FASSUNGEN
low cost ..... pro Pin: 0.02
Präzision ..... pro Pin: 0.05
Fordern Sie unser Elektronik-Info an. Wir haben Preise, wie sie sonst nur Großhändler Kunden gewähren!!!

MONITORE
COMMODORE 1801
Farbmonitor 559.00
ORION-Farb-Monitor 1280 ..... 798.00
GRÜNMONITOR
mit Ton 189.00
Universalmodem
für C 64 ..... 229.00
für ATARI XL 249.-
für IBM 249.-

JOYSTICKS
QUICKSHOT II 13.90
QUICKSHOT IX 24.90
QUICKGUN III 29.90
COMPETITIONpro 27.90
DELA-MAUS C 64 ..... 99.00

EPROMKARTEN C 64
256 K Superepromkarte ohne Gehäuse 109.00
64 K Superepromkarte mit Gehäuse 49.50
Epromkarte 2.1 ..... 14.00
Eprommer II mit Gehäuse 149.00
64K-RAM-Floppy 149.00
5-fach Betriebssystemkarte mit Schalter 39.90
IC-Tester 134.00
Hardcopy-Modul 49.00
Steckplatz-erweiterung 99.00
DELA CPM-Modul 169.00
Relaiskarte 98.00

ZUBEHÖR
ATARI ST Doppelkopf-laufwerk 498.-
Lochraster-Experimentier-karte 13.90

Nachnahmeversand bei Auftragswert ab 30.– DM. Unter DM 30.– Vorkasse mittels V-Scheck oder über Postscheck. Auslandsversand nur gegen Vorkasse. 15.– DM Versandkosten. Bestellungen (soweit vorrätig) bis 12 Uhr werden am selben Tag verschickt. Wir liefern Ihnen zu den Verkaufs- und Lieferbedingungen des Elektronikgewerbes. Fordern Sie unser kostenloses Elektronik-Info an!

● Fordern Sie unseren kostenlosen, farbigen Katalog an!

NEU! NEU! NEU! NEU!
24 Stunden Bestellservice
Unter der Rufnummer
0221/517081
stehen wir Ihnen von Montag-Freitag in der Zeit von 9-18 h für alle Bestellungen/Fragen zur Verfügung.
Unter der Rufnummer
0221/517084
steht Ihnen unser Anrufbeantworter täglich 24 Std. zur Verfügung.

NEU! NEU! NEU! NEU!
DELA-ESSEN
Schützenbahn 11-13
(DGB-Haus, Porscheplatz)

DELA Elektronik
Maastrichter Str. 23 · 5000 Köln 1
☎ 0221/517081

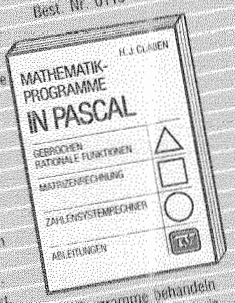
WI 1.4

HEISE/LUTHER
Bissendorfer Straße 8
3000 Hannover 61

Es geht um Anwendungen aus der Mathematik, Physik und Chemie. Die ausführliche Darstellung der mathematischen Grundlagen wird durch Zahlenbeispiele so demonstriert, daß auch der Praktiker, der ohne Mathematik auskommen möchte, einen guten Zugang findet.
Best. Nr. 7008-9
DM 59,80

Dieses Buch enthält acht BASIC-Programme aus den Fachbereichen Mathematik, Mechanik und Elektrotechnik. Es können z. B. Trafo- und Biegeträgerberechnungen sowie Torsions- und Biegebelastung bei Motor und Getriebeteilen berechnet werden.
Best. Nr. 0115-4 DM 19,80

Eine Softwarebibliothek mit 112 Pascalprogrammen. Der Schwerpunkt liegt im mathematischen und elektrotechnischen Bereich. Anfänger finden hier einen sehr praxisbezogenen Zugang zu Pascal. Profis eine sofort einsetzbare Programmbibliothek.
Best. Nr. 9102-8
DM 49,80



Dem mathematisch nicht speziell ausgebildeten Praktiker soll ein Zugang zu den gebräuchlichsten statistischen Verfahren eröffnet werden. Das Buch stellt 30 Programme zur Verfügung. Jedes ist ausführlich dokumentiert und mit Hinweisen zur Bedienung und Interpretation der Ergebnisse versehen.
Best. Nr. 0128-6
DM 39,80

Die Programme behandeln numerische Probleme, wie z. B. das Ableiten beliebiger Funktionen. Darüber hinaus ist auch ein „Rechner" vorhanden, der beliebige Zahlensysteme verarbeiten kann.
Best. Nr. 0114-6 DM 39,80

Sollten unsere Bücher und Softwarepakete nicht bei Ihrem Fachhändler erhältlich sein, bitte direkt anfordern und Verrechnungsscheck zzgl. DM 3,50 Versandkostenpauschale beifügen.

Gleichzeitig editieren und kompilieren

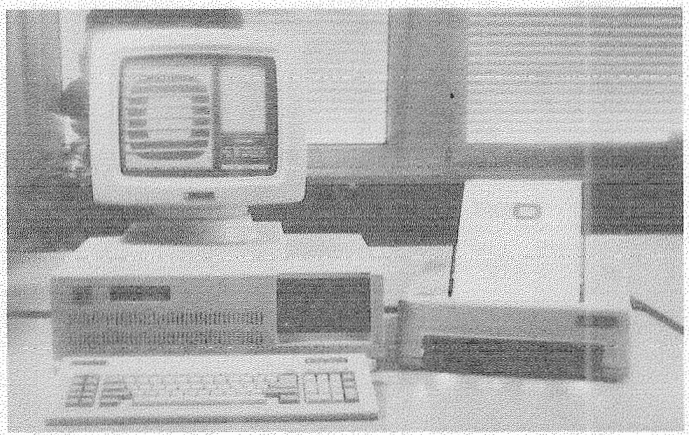
Double-DOS macht den Traum vieler Softwareentwickler wahr, gleichzeitig unter MS-DOS editieren und kompilieren zu können. Das Multi-Tasking-Programm verlangt vom Anwender keinen zusätzlichen Bedienungsaufwand, ermöglicht aber eine wesentlich bessere Auslastung des Rechners und eine höhere Arbeitsproduktivität. Programme werden im Hintergrund kompiliert, während im Vordergrund weitere Programme editiert werden können. Mit einem Tastendruck kann vom Vorder- zum Hintergrund gewechselt werden. Double-DOS unterstützt zahlreiche bekannte Anwendungen von Textverarbeitung über Tabellenkalkulation bis hin zu Datenbank-Applikationen. Das Programm kostet rund 270 DM.

AiD, Ingenieurgesellschaft für Mikrocomputer Hard- und Software mbH
Bauernfeindstr. 55 d
8500 Nürnberg 50
☎ 09 11/43 70 67

Dem Computer das Sehen beibringen

Für den Betrieb am Macintosh und an Osborne- und IBM-PC-kompatiblen Computern wurden die Scanner der Serie MS 200/200A konzipiert. Entsprechend der eingesetzten Software eröffnen sich verschiedene Anwendungen: Mit dem Programmpaket Eyestar werden Bilder in den Computer mit einer Auflösung von 300 Punkten pro Zoll übernommen. Kontrast, Helligkeit, Grautonselektierung und Normierung lassen sich dabei speziell einstellen. Mit dem Programm Read Right wird der Scanner zu einem OCR-Lesegerät und praktiziert damit die Textübernahme. Von der Software her ist dabei eine große Bandbreite von Schriftarten und Punktgrößen zugelassen – sogar eine Mischung verschiedener Schriften auf einem Blatt Papier bringt das Gespann aus Scanner und Computer nicht durcheinander. Voraussetzung zum Einsatz des Systems ist ein leidlich sauberes Schriftbild. Eigene Interpretations- und Schätzungsfunktionen, wie sie beispielsweise das menschliche Gehirn bei Lücken im optischen Bild anwendet, fehlen einem solchen System natürlich.

Osborne Computer
Dingolfinger Str. 6
8000 München 80
☎ 0 89/49 10 01



Der Osborne-Scanner (rechts) dient zum Lesen von Schrift und Erfassen von Zeichnungen

Multi-User-Betrieb unter MS-DOS

MULTILink Advanced 4.0, eine reine Softwarelösung für XT- und AT-kompatible Rechner, die einen echten Multi-User-Betrieb unter PC/MS-DOS erlaubt, ist jetzt auch in Deutschland erhältlich. Gleichartige Terminals werden über V.24-Schnittstellen miteinander verbunden. Die Anpassung kann in wenigen Minuten erfolgen, wobei sich Arbeitsweise und Befehle des DOS nicht verändern.

In der neuen Version werden auch alle Grafikfähigkeiten der angeschlossenen PCs im Terminalmodus unterstützt. Die Anschlußmöglichkeiten wurden derart erweitert, daß bis zu 16 Terminals (denen jeweils bis zu 640 KByte im Hauptspeicher zugeordnet werden können) anschließbar sind. Durch eine höhere Übertragungsgeschwindigkeit zu den Terminals – maximal 115 200 Bd – bringen nun auch schnelle Speicherzugriffe keine zeitlichen Hürden mehr ins System. Für den Programmierer stellt schließlich „Supercom" eine Unterstützung bei der Kommunikation dar.

Wer eine ältere MULTILink-Version besitzt, kann gegen Einsendung des Originalpaketes zu einem Preis von rund 500 DM eine Aufrüstung beanspruchen; eine komplette Neuversion kostet knapp über 2300 DM.

Compucon Deutschland GmbH
Jahnstr. 22
8037 Olching
☎ *0 81 42/2 80 41*

Preislich recht attraktives Hardware/Software-Komplettpaket für Handwerk und mittelständische Industrie: Zenith-PC, Drucker und modulare ASSI-Software

Interessantes Paket für Einsteiger

Vom Preis/Leistungs-Verhältnis her beachtlich ist ein für das Handwerk und die mittelständische Industrie zusammengestelltes Angebotspaket, das ein Computer-Komplettsystem einschließlich Software auf den Tisch des Kunden stellt. Das System wird dem Motto „auspacken und gleich sinnvoll arbeiten" gerecht. Für 5500 DM erhält man einen

Schaltungsideen für Elektroniker

NEU
Elektronik-Sonderheftserie „Schaltungspraxis"

Arbeitserleichterungen, die es in sich haben, weil sie Entwicklungszeit und damit Kosten sparen: Die interessantesten Applikationsbeispiele aus den ELEKTRONIK-Rubriken „Schaltungspraxis", „µC-Praxis" und „Applikation" finden Sie jetzt in einer neuen Sonderheftserie. Thematisch geordnet sind sie eine einzigartige Arbeitshilfe für Ingenieure und Elektroniker. **Die einzelnen Hefte sind unter folgenden Titeln erschienen:** ● Mikroprozessorschaltungen ● Steuer- und Regelschaltungen ● Meß-, Prüf- und Sensorschaltungen ● Interface- und Wandler-Schaltungen ● Digital- und Logikschaltungen ● Oszillatoren, Generatoren, Rechen- und Filterschaltungen. ● Analog-, Operationsverstärker, Stromversorgung.

Nr. 7: Analog-, Operationsverstärker, Stromversorgung

● Beispiele für schnelle Schalter-Anwendungen, für die Ansteuerung von Leistungstransistoren sowie für Operations- und Instrumentenverstärker. Weitere Applikationen beschreiben Netz- und Ladegeräte, Schaltregler und Schaltnetzteile.

Hier gibt's die neuen Arbeitshilfen für Elektroniker:

Bei allen Bahnhofsbuchhandlungen, beim Elektronik-Fachhandel, in Mikrocomputershops, bei größeren Zeitschriftenverkaufsstellen, in Buchhandlungen oder direkt beim Franzis-Verlag

● Per Postkarte mit Hinweis „Sonderheft Analog-, Operationsverstärker, Stromversorgung" (Rechnung wird zugesandt) oder

● Per Scheck (DM 22.50)

Franzis'
Franzis-Verlag, Karlstraße 37,
8000 München 2,
Tel. 0 89/51 17-2 39/-3 80

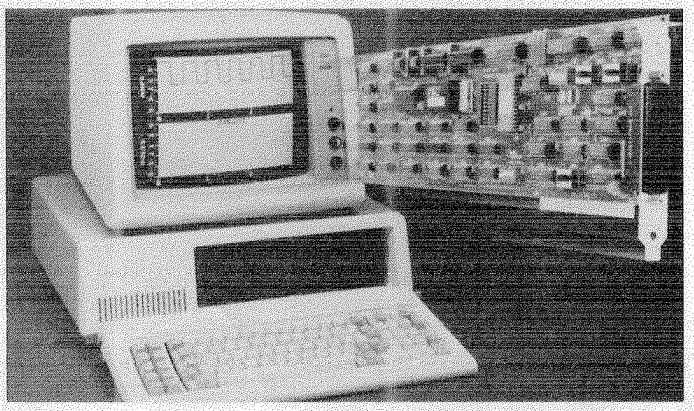
In der Schweiz: Thali AG, Fachliteratur und Software, Industriestraße 6,
CH-6285 Hitzkirch

In Österreich: Erb-Verlag,
Ges.m.b.H. & Co. KG, Mariahilfer Straße 71,
A-1061 Wien

Zenith-PC Z-148 mit zwei Diskettenlaufwerken und einem Monochrom-Bildschirm, einen Matrixdrucker LX-86 (80 Zeichen/s) sowie die Grundkonfiguration der Handwerker-Software ASSI. Diese besteht aus den Modulen Angebotskalkulation und -druck, Auftragsbestätigung, Faktura, Leistungs-, Kunden- und Stammdatenverwaltung. Gegen Aufpreis kann man die Software bausteinartig erweitern, z. B. um spezielle Elektro-Programmteile, um Module zur Preiskalkulation, Personalverwaltung oder Warenwirtschaft – doch mit der Einstiegs-Konfiguration kann man auch schon eine ganze Menge anfangen. Wer seinem Computer bzw. sich selbst den Komfort einer 20-MByte-Festplatte gönnen möchte, muß rund 1 500 DM zusätzlich einplanen.

Zenith Data Systems GmbH
Postfach 10 20 60
6072 Dreieich
☎ 0 61 03/3 90 50

Sorgt für schnelle Kontakte zur Außenwelt



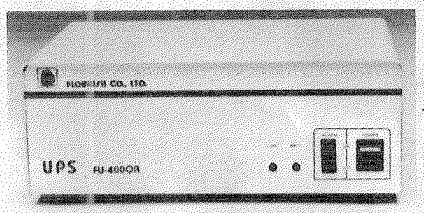
DMA-Routinen sorgen in der Analog-Interfaceplatine Dash-16F für Schnelligkeit

Als Analog-Interface mit einer maximalen Abtastfrequenz von 100 kHz präsentiert sich die Dash-16F-Platine von Metrabyte dem an der Erfassung, Verarbeitung und Ausgabe von Analogsignalen interessierten PC-Benutzer. Mit einer Wandler-Auflösung von 12 Bit stehen 16 unsymmetrische oder acht Differenzeingänge zur Verfügung; die gleiche Auflösung wird von den beiden Analogausgängen angeboten. Steuersignale können über acht digitale E/A-Leitungen laufen; die Kontrolle sämtlicher Erfassungs- und Ausgabezyklen übernimmt ein programmierbarer Zähler/Zeitgeber. Im Lieferumfang enthalten sind ein Softwaretreiber für das Arbeiten mit interpretiertem oder compiliertem Basic und einige Beispiel- und Kalibrier-Routinen. Treiber für Turbo-Pascal, Fortran und C sind optional. Im Gegensatz zu Platinen, die nur in AT-Computern einsetzbar sind, ist diese Baugruppe in sämtlichen PC-Varianten verwendbar.

Keithley Instruments GmbH
Heiglhofstr. 5
8000 München 70
☎ 0 89/71 00 20

Unterbrechungsfreie Stromversorgungen für Computer (USV) erstmalig unter der Abschreibungsgrenze!

z.B. USV 200



DM 698,-

- für alle gängigen PC's geeignet
- eingebaute Batterie
- Pufferzeit ca. 20 Minuten
- Ausgangsspannung trapezförmig

Einzelstückpreise:

USV 200 698,— DM USV 400 890,— DM
USV 300 798,— DM USV 500 992,— DM

Power Components GmbH
Max-Planck-Str. 11 · 8910 Landsberg/Lech
☎ 08191 - 4042 · Telex 527153 pocom d

NoName PC — Endlich unter 1000,— DM ..... 999,—
mit deutscher Tastatur, 256 KB Hauptspeicher, 360 KB Laufwerk, Color-Graphik, Netzteil, VOLLKOMPATIBEL, VOLLAUFRÜSTBAR

NoName PC (— Die Alternative —)..... 1666,—
Grundgerät + Monitor + Maus + 512 KB + Multi-IO Card + Turbo-Erweiterung (8 MHz) + V20 CPU

NoName XT · 20 MB Vollausbau ..... 2999,—
20 MB Festplatte, 150 Watt Netzteil, 12"TTL-Monitor, Hercules-Karte, Dislaufwerk, Multi-IO, 640 KB, V20 CPU

NoName AT · Preis Sensation ..... 2222,—
Super Tastatur, 512 KB, Color Graphik, Slimeline 8 MHz Modus!

20 MB Festplatten Aufpreis ..... 1555,—

ZUBEHÖR PREISSENKUNGEN

20 MB Festplatte + Controller	**1111,—**
100 Disketten ..... nur noch	**99,—**
V-20 CPU	**25,—**
Joystick	**49,—**
Multifunktion incl. 384 KB, Uhr, Schnittstellen	**399,—**
8087 Coprozessor	**377,—**
MS kompatible Mouse	**199,—**
1,2 MB Disklaufwerk und Controller für XT	**555,—**
20 High Quality 3 1/2" Festplatte + Controller	**1333,—**
30 MB High Quality 3 1/2" Festplatte + Controller	**1599,—**
HQ-Disk mit 12 Monaten Garantie, sehr schnell und superleise!	
Aufpreis für Harddisk Einsteckkarte	**199,—**

Z + M EDV-Büro GmbH,
Schloßstr. 69,
1000 Berlin 41
☎ (030) 834 88 55 / 859 20 36

Wer COMMODORE kauft, muß an VOBIS denken:

VON AMIGA BIS ZUBEHÖR!

COMMODORE C 16

Einsteigercomputer
Mit 16 K, Farbe, Grafik, Sound

125.-

Daten-Recorder dazu **59.-**

BASIC-Lernkurs auf Kassette 10.-
C 16 Buch 10.-

Weitere COMMODORE-Preise

C 116 mit 64 K-Speicher	**149.-**
C 64 II incl. GEOS-Diskette	**475.-**
1541 Floppy für C 64 II	**475.-**
128 D Komplettcomputer	**1248.-**
SUPERSCRIPT für 128 Textprogramm	**99.-**
SUPERBASE für 128 Kalkulation	**99.-**

AMIGA

Basispaket	**1698.-**
1081 RGBI-Farb-Monitor dazu	**875.-**
GrafiCraft + TextCraft für AMIGA	**95.-**
MS-DOS Disk für AMIGA	**185.-**
Komplett	**2698.-**

Sie sparen 155.- DM im Vergleich zu den Einzelpreisen!

1520 4-farb Drucker/Plotter

für C 116, C 16 VC 20, C 64, C 128 (D)

anschlußfertig

199.-

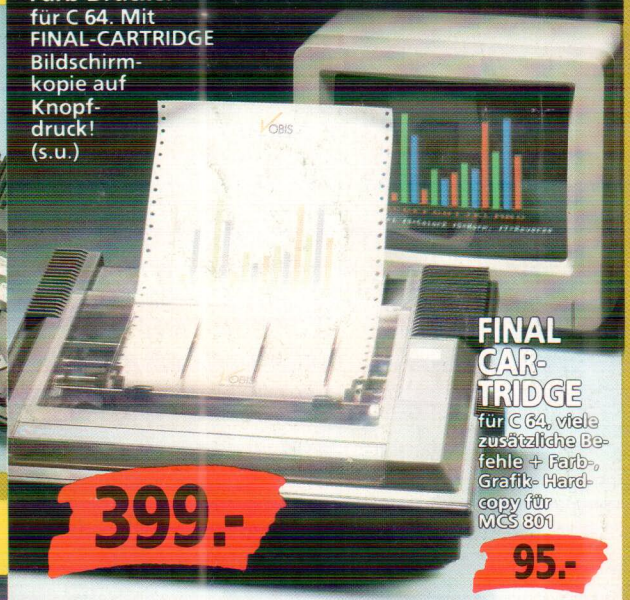
NEU! MCS 801

Farb-Drucker für C 64. Mit FINAL-CARTRIDGE Bildschirmkopie auf Knopfdruck! (s.u.)

1901-RGBI-Farbmonitor

für C 64, 128, PC 10/20

698.-



399.-

FINAL CARTRIDGE

für C 64, viele zusätzliche Befehle + Farb-, Grafik-Hardcopy für MCS 801

95.-

Universell: PLUS 4

Grafik, Farbe, Sound — 64 K Speicher

Incl. 4 Anwenderprogammen: Text, Datei, Kalkualtion, Grafik

198.-

COMMODORE MPS 1000

Universeller Matrixdrucker

NLQ-Schönschrift
Traktor + Einzelblätter
2 Schnittstellen: COMMODORE VC + IBM-PC oder Kompatible

675.-

CITIZEN-DRUCKER IDP-560

2-farb Matrixdrucker

40 Zeichen pro Zeile, alle COMMODORE VC-Zeichen

Anschlußfertig für COMMODORE VC 20, C 16, C 116, PLUS/4, C 64 und C 128

nur **99.-**

MONITORE:

HIGHSCREEN DP 602-P 31
22 Mhz, 12", grün mit Ton **199.-**

HIGHSCREEN CM 95 RGBI-Farbmonitor für C 64, 128, PC's o. ATARI ST (Scart) nur **675.-**

▼Ausfüllen, ausschneiden und ab die Post an **VOBIS**, Postfach 1778, **5100 Aachen**

TIP

COMMODORE PC 10-II
640KRam, AGA-Grafikkarte, 2 Disk-Laufwerke à 360 K. IBM-kompatibel. Jetzt nur noch **2775.-**

PC 10-II
wie oben, jedoch zusätzlich mit 20 MB Festspeicherplatte
Jetzt nur noch **3775.-**

1551 — 170 K-Floppy

Für CMMODORE C 116, C 16 und PLUS 4

komplett anschlußfertig

299.-

Ich will mehr wissen!
Senden Sie mir:
☐ Prospekte ☐ Preisliste
Meine Adresse _____

kompetent + preiswert

WICHTIG! Herstellerbedingte Lieferzeiten! Aufgrund erhöhter Nachfrage ist nicht immer alles sofort lieferbar!

VOBIS
Deutschlands umsatzgrößter Microcomputer-Spezialist

VERSAND-ZENTRALE:
Postfach 1778
Rotter Bruch 32-34
5100 AACHEN
☎ 0241/50 00 81
Tx 832389 vobis d

FILIALEN:

AACHEN Viktoriastr. 74 · 0241/54 31 00
AACHEN Pontstraße 60 · 0241/3 38 06
BERLIN 30 Kurfürstenstr. 101 · 030/2 13 94 80
BIELEFELD Herforder Str. 106 · 0521/6 38 78
BREMEN Violenstraße 37 · 0421/32 04 20
DORTMUND Hamburger Str. 110 · 0231/57 30 72
DÜSSELDORF Wielandstr. 21 · 0211/35 99 64
FRANKFURT Frankenallee 207/209 · 069/73 40 49
HAMBURG Krohnskamp 15 · 040/2 79 46 76
HANNOVER Berliner Allee 47 · 0511/81 65 71
KARLSRUHE Kriegsstr. 27/29 (BGH) 0721/37 82
KONSTANZ Kreuzlinger Str. 18 · 07531/1 55 6
KÖLN Mathiasstr. 24-26 · 0221/24 86 42
MÜNCHEN Aberlestr. 3 · 089/77 21 10
NÜRNBERG Vord. Ledergasse 8 · 0911/23 29 9
STUTTGART Marienstr. 11-13 · 0711/60 63 36